INTERNATIONAL TABLES for CRYSTALLOGRAPHY

Volume

G

Definition and exchange of crystallographic data

Edited by
Sydney Hall and
Brian McMahon

First edition

INTERNATIONAL TABLES FOR CRYSTALLOGRAPHY

International Tables for Crystallography

Volume A: *Space-Group Symmetry*
Editor Theo Hahn
First Edition 1983, Fifth Edition 2002
Corrected Reprint 2005

Volume A1: *Symmetry Relations between Space Groups*
Editors Hans Wondratschek and Ulrich Müller
First Edition 2004

Volume B: *Reciprocal Space*
Editor U. Shmueli
First Edition 1993, Second Edition 2001

Volume C: *Mathematical, Physical and Chemical Tables*
Editor E. Prince
First Edition 1992, Third Edition 2004

Volume D: *Physical Properties of Crystals*
Editor A. Authier
First Edition 2003

Volume E: *Subperiodic Groups*
Editors V. Kopský and D. B. Litvin
First Edition 2002

Volume F: *Crystallography of Biological Macromolecules*
Editors Michael G. Rossmann and Eddy Arnold
First Edition 2001

Volume G: *Definition and Exchange of Crystallographic Data*
Editors Sydney Hall and Brian McMahon
First Edition 2005

INTERNATIONAL TABLES FOR CRYSTALLOGRAPHY

Volume G

DEFINITION AND EXCHANGE
OF CRYSTALLOGRAPHIC DATA

Edited by

SYDNEY HALL AND BRIAN MCMAHON

First edition

Published for

THE INTERNATIONAL UNION OF CRYSTALLOGRAPHY

by

SPRINGER

2005

A C.I.P. Catalogue record for this book
is available from the Library of Congress
ISBN 0-4020-3138-6 (acid-free paper)
ISSN 1574-8707

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands

Sold and distributed in North, Central and South America
by Springer,
101 Philip Drive, Norwell, MA 02061, USA

In all other countries, sold and distributed
by Springer,
P.O. Box 322, 3300 AH Dordrecht, The Netherlands

Technical Editor: N. J. Ashcroft

© International Union of Crystallography 2005

Short extracts may be
reproduced without formality, provided that the
source is acknowledged, but substantial portions
may not be reproduced by any process
without written permission from the
International Union of Crystallography

Printed in Denmark by P. J. Schmidt A/S

Associate editors

H. M. Berman
F. C. Bernstein
H. J. Bernstein
P. E. Bourne
I. D. Brown
P. M. D. Fitzgerald
R. W. Grosse-Kunstleve
J. R. Hester
N. Spadaccini
B. H. Toby
J. D. Westbrook

Contributing authors

F. H. Allen, Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge, CB2 1EZ, England. [2.4, 4.1, 4.8]

J. M. Barnard, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England. [2.4, 4.8]

H. M. Berman, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA. [2.6, 3.6, 4.5, 5.5]

H. J. Bernstein, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA. [2.2.7, 2.3, 3.7, 4.6, 5.1, 5.4, 5.6]

P. E. Bourne, Research Collaboratory for Structural Bioinformatics, San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0537, USA. [3.6, 4.5]

I. D. Brown, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1. [2.2.7, 3.5, 3.8, 4.1, 4.7]

A. P. F. Cook, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England. [2.4, 2.5, 4.8]

P. J. Ellis, Stanford Linear Accelerator Center, 2575 Sand Hill Road, Menlo Park, CA 94025, USA. [5.6]

Z. Feng, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA. [5.5]

P. M. D. Fitzgerald, Merck Research Laboratories, Rahway, New Jersey, USA. [3.2, 3.6, 4.5]

S. R. Hall, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia. [1.1, 2.1, 2.2, 2.4, 2.5, 2.6, 3.2, 4.1, 4.8, 4.9, 4.10, 5.2, 5.4]

A. P. Hammersley, ESRF/EMBL Grenoble, 6 rue Jules Horowitz, France. [2.3, 4.6]

K. Henrick, EMBL Outstation, The European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, England. [Appendix 3.6.2]

M. A. Hoyland, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England. [5.7]

B. McMahon, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England. [1.1, 2.2.7, 3.1, 3.2, 3.6, 4.5, 5.2, 5.3, 5.7]

G. Madariaga, Departamento de Física de la Materia Condensada, Facultad de Ciencia y Tecnología, Universidad del País Vasco, Apartado 644, 48080 Bilbao, Spain. [3.4, 4.3]

P. R. Mallinson, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, Scotland. [3.5, 4.4]

N. Spadaccini, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia. [2.1, 2.2.7, 5.2]

P. R. Strickland, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England. [5.7]

B. H. Toby, NIST Center for Neutron Research, National Institute of Standards and Technology, Gaithersburg, Maryland 20899-8562, USA. [3.3, 4.2]

K. D. Watenpaugh, retired; formerly Structural, Analytical and Medicinal Chemistry, Pharmacia Corporation, Kalamazoo, Michigan, USA. [3.6, 4.5]

J. D. Westbrook, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA. [2.2, 2.6, 3.6, 4.5, 4.6, 4.10, 5.5]

E. L. Ulrich, Department of Biochemistry, University of Wisconsin Madison, 433 Babcock Drive, Madison, WI 53706-1544, USA. [Appendix 3.6.2]

H. Yang, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA. [5.5]

Contents

PAGE

Preface (S. R. Hall and B. McMahon) xi

PART 1. HISTORICAL INTRODUCTION

1.1. Genesis of the Crystallographic Information File (S. R. Hall and B. McMahon) 2
1.1.1. Prologue 2
1.1.2. Past approaches to data exchange 2
1.1.3. Card-image formats 3
1.1.4. The Standard Crystallographic File Structure (SCFS) 3
1.1.5. The impact of networking on crystallography 4
1.1.6. The Working Party on Crystallographic Information (WPCI) 5
1.1.7. The Crystallographic Information File 6
1.1.8. Diversification: the Molecular Information File and dictionary definition language 7
1.1.9. The macromolecular Crystallographic Information File 7
1.1.10. The Crystallographic Binary File 8
1.1.11. Other extension dictionaries 8
1.1.12. The broader context: CIF and XML 9
References 9

PART 2. CONCEPTS AND SPECIFICATIONS

2.1. Specification of the STAR File (S. R. Hall and N. Spadaccini) 13
2.1.1. Introduction 13
2.1.2. Universal data language concepts 13
2.1.3. The syntax of the STAR File 14
Appendix 2.1.1. Backus–Naur form of the STAR syntax and grammar 16
References 19

2.2. Specification of the Crystallographic Information File (CIF) (S. R. Hall and J. D. Westbrook) 20
2.2.1. Introduction 20
2.2.2. Terminology 20
2.2.3. The syntax of a CIF 21
2.2.4. Portability and archival issues 22
2.2.5. Common semantic features 23
2.2.6. CIF metadata and dictionary compliance 25
2.2.7. Formal specification of the Crystallographic Information File (S. R. Hall, N. Spadaccini, I. D. Brown, H. J. Bernstein, J. D. Westbrook and B. McMahon) 25
References 36

2.3. Specification of the Crystallographic Binary File (CBF/imgCIF) (H. J. Bernstein and A. P. Hammersley) 37
2.3.1. Introduction 37
2.3.2. CBF and imgCIF 37
2.3.3. Overview of the format 39
2.3.4. A complex example 42
2.3.5. imgCIF encodings 42
Appendix 2.3.1. Deprecated CBF conventions 43
References 43

2.4. Specification of the Molecular Information File (MIF) (F. H. Allen, J. M. Barnard, A. P. F. Cook and S. R. Hall) 44
2.4.1. Introduction 44

2.4.2. Historical background ... 44
2.4.3. MIF objectives ... 45
2.4.4. MIF concepts and syntax ... 45
2.4.5. Atoms, bonds and molecular representations ... 47
2.4.6. Bonding conventions ... 48
2.4.7. Structural templates ... 48
2.4.8. Stereochemistry and geometry at stereogenic centres ... 48
2.4.9. MIF query applications ... 50
2.4.10. Conclusion ... 51
References ... 51

2.5. Specification of the core CIF dictionary definition language (DDL1) (S. R. Hall and A. P. F. Cook) ... 53
2.5.1. Introduction ... 53
2.5.2. The organization of a CIF dictionary ... 53
2.5.3. Definition attributes ... 54
2.5.4. DDL versions ... 54
2.5.5. The structure of DDL1 definitions ... 55
2.5.6. DDL1 attribute descriptions ... 57
References ... 60

2.6. Specification of a relational dictionary definition language (DDL2) (J. D. Westbrook, H. M. Berman and S. R. Hall) ... 61
2.6.1. Introduction ... 61
2.6.2. The DDL2 presentation ... 61
2.6.3. Overview of the elements of DDL2 ... 62
2.6.4. DDL2 organization ... 63
2.6.5. DDL2 dictionary applications ... 63
2.6.6. Detailed DDL2 specifications ... 65
References ... 70

PART 3. CIF DATA DEFINITION AND CLASSIFICATION

3.1. General considerations when defining a CIF data item (B. McMahon) ... 73
3.1.1. Introduction ... 73
3.1.2. Informal definition procedures ... 74
3.1.3. Formal definition process ... 74
3.1.4. Choice of data model ... 75
3.1.5. Constructing a DDL1 dictionary ... 76
3.1.6. Constructing a DDL2 dictionary ... 79
3.1.7. Composing new data definitions ... 83
3.1.8. Management of multiple dictionaries ... 85
3.1.9. Composite dictionaries ... 88
3.1.10. Public CIF dictionaries ... 89
References ... 91

3.2. Classification and use of core data (S. R. Hall, P. M. D. Fitzgerald and B. McMahon) ... 92
3.2.1. Introduction ... 92
3.2.2. Experimental measurements ... 92
3.2.3. Analysis ... 98
3.2.4. Atomicity, chemistry and structure ... 102
3.2.5. Publication ... 112
3.2.6. File metadata ... 114

Appendix 3.2.1. Category structure of the core CIF dictionary . . . 116

References . . . 116

3.3. Classification and use of powder diffraction data (B. H. Toby) . . . 117

3.3.1. Introduction . . . 117

3.3.2. Dictionary design considerations . . . 117

3.3.3. pdCIF dictionary sections . . . 117

3.3.4. Experimental measurements . . . 118

3.3.5. Analysis . . . 121

3.3.6. Atomicity, chemistry and structure . . . 123

3.3.7. File metadata . . . 123

3.3.8. pdCIF for storing unprocessed measurements . . . 126

3.3.9. Use of pdCIF for Rietveld refinement results . . . 128

3.3.10. Other pdCIF applications . . . 129

Appendix 3.3.1. Category structure of the powder CIF dictionary . . . 130

References . . . 130

3.4. Classification and use of modulated and composite structures data (G. Madariaga) . . . 131

3.4.1. Introduction . . . 131

3.4.2. Dictionary design considerations . . . 131

3.4.3. Arrangement of the dictionary . . . 132

3.4.4. Use of the msCIF dictionary . . . 137

Appendix 3.4.1. Category structure of the msCIF dictionary . . . 139

References . . . 139

3.5. Classification and use of electron density data (P. R. Mallinson and I. D. Brown) . . . 141

3.5.1. Introduction . . . 141

3.5.2. Dictionary design considerations . . . 141

3.5.3. Classification of data definitions . . . 141

3.5.4. Development of the dictionary and supporting software . . . 143

References . . . 143

3.6. Classification and use of macromolecular data (P. M. D. Fitzgerald, J. D. Westbrook, P. E. Bourne, B. McMahon, K. D. Watenpaugh and H. M. Berman) . . . 144

3.6.1. Introduction . . . 144

3.6.2. Considerations underlying the design of the dictionary . . . 144

3.6.3. Overview of the mmCIF data model . . . 145

3.6.4. Content of the macromolecular CIF dictionary . . . 147

3.6.5. Experimental measurements . . . 148

3.6.6. Analysis . . . 152

3.6.7. Atomicity, chemistry and structure . . . 164

3.6.8. Publication . . . 190

3.6.9. File metadata . . . 194

Appendix 3.6.1. Category structure of the mmCIF dictionary . . . 195

Appendix 3.6.2. The Protein Data Bank exchange data dictionary (J. D. Westbrook, K. Henrick, E. L. Ulrich and H. M. Berman) . . . 195

References . . . 198

3.7. Classification and use of image data (H. J. Bernstein) . . . 199

3.7.1. Introduction . . . 199

3.7.2. Binary image data . . . 199

3.7.3. Axes ... 201
3.7.4. The diffraction experiment ... 202
Appendix 3.7.1. Category structure of the CBF/imgCIF dictionary ... 205
References ... 205

3.8. Classification and use of symmetry data (I. D. Brown) ... 206
3.8.1. Introduction ... 206
3.8.2. Dictionary design considerations ... 206
3.8.3. Arrangement of the dictionary ... 207
3.8.4. Future developments ... 208
References ... 208

PART 4. DATA DICTIONARIES

4.1. Core dictionary (coreCIF) (S. R. Hall, F. H. Allen and I. D. Brown) ... 210

4.2. Powder dictionary (pdCIF) (B. H. Toby) ... 258

4.3. Modulated and composite structures dictionary (msCIF) (G. Madariaga) ... 270

4.4. Electron density dictionary (rhoCIF) (P. R. Mallinson) ... 290

4.5. Macromolecular dictionary (mmCIF) (P. M. D. Fitzgerald, J. D. Westbrook, P. E. Bourne, B. McMahon, K. D. Watenpaugh and H. M. Berman) ... 295

4.6. Image dictionary (imgCIF) (A. P. Hammersley, H. J. Bernstein and J. D. Westbrook) ... 444

4.7. Symmetry dictionary (symCIF) (I. D. Brown) ... 459

4.8. Molecular Information File dictionary (MIF) (F. H. Allen, J. M. Barnard, A. P. F. Cook and S. R. Hall) ... 467

4.9. DDL1 dictionary (S. R. Hall) ... 471

4.10. DDL2 dictionary (J. D. Westbrook and S. R. Hall) ... 473

PART 5. APPLICATIONS

5.1. General considerations in programming CIF applications (H. J. Bernstein) ... 481
5.1.1. Introduction ... 481
5.1.2. Background ... 481
5.1.3. Strategies in designing a CIF-aware application ... 483
5.1.4. Conclusion ... 486
References ... 486

5.2. STAR File utilities (N. Spadaccini, S. R. Hall and B. McMahon) ... 488
5.2.1. Introduction ... 488
5.2.2. Data instances and context ... 488
5.2.3. *Star_Base*: a general-purpose data extractor for STAR Files ... 491
5.2.4. Editing STAR Files with *Star.vim* ... 494
5.2.5. Browser-based viewing with *StarMarkUp* ... 494
5.2.6. Object-oriented STAR programming ... 495
References ... 497

5.3. Syntactic utilities for CIF (B. McMahon) ... 499
5.3.1. Introduction ... 499
5.3.2. Syntax checker ... 499
5.3.3. Editors with graphical user interfaces ... 501

5.3.4. Data-name validation ... 507
5.3.5. File transformation software ... 509
5.3.6. Libraries for scripting languages ... 515
5.3.7. Rapid development tools ... 517
5.3.8. Tools for mmCIF ... 522
5.3.9. Concluding remarks ... 525
References ... 525

5.4. *CIFtbx*: Fortran tools for manipulating CIFs (H. J. Bernstein and S. R. Hall) ... 526
5.4.1. Introduction ... 526
5.4.2. An overview of the library ... 526
5.4.3. Initialization commands ... 526
5.4.4. Read commands ... 527
5.4.5. Write commands ... 527
5.4.6. Variables ... 528
5.4.7. Name aliases ... 529
5.4.8. Implementation of the tools ... 530
5.4.9. How to read CIF data ... 530
5.4.10. How to write a CIF ... 530
5.4.11. Error-message glossary ... 531
5.4.12. Internals and programming style ... 535
5.4.13. Distribution ... 538
References ... 538

5.5. The use of mmCIF architecture for PDB data management (J. D. Westbrook, H. Yang, Z. Feng and H. M. Berman) ... 539
5.5.1. Introduction ... 539
5.5.2. Representing macromolecular structure data ... 539
5.5.3. Integrated data-processing system: overview ... 541
5.5.4. Access ... 543
References ... 543

5.6. *CBFlib*: an ANSI C library for manipulating image data (P. J. Ellis and H. J. Bernstein) ... 544
5.6.1. Introduction ... 544
5.6.2. *CBFlib* function descriptions ... 545
5.6.3. Compression schemes ... 552
5.6.4. Sample templates ... 552
5.6.5. Example programs ... 555
References ... 556

5.7. Small-molecule crystal structure publication using CIF (P. R. Strickland, M. A. Hoyland and B. McMahon) ... 557
5.7.1. Introduction ... 557
5.7.2. Case study: the fully automated reporting of small-unit-cell crystal structures ... 557
5.7.3. CIF and other journals ... 562
Appendix 5.7.1. Request list for *Acta Crystallographica Section C* ... 564
Appendix 5.7.2. Data validation using *checkcif* ... 566
References ... 568

Subject index ... 571

Index of categories and data names ... 578

Preface

By S. R. Hall and B. McMahon

Crystallography is a data-intensive discipline. This is true of the measurement and the analytical aspects of crystallographic studies, and so the orderly acquisition and retention of data is of great importance. There is also a need for computational tools that facilitate efficient data-handling processes. In this data-rich environment, *International Tables for Crystallography* provides supporting sources of reference data and guidelines for analysing, interpreting and modelling the data involved in a wide variety of studies.

This volume in the *International Tables* series, *Definition and exchange of crystallographic data*, deals with the precise definition of data items used in crystallography. It focuses on a particular data representation, the Crystallographic Information File (CIF), developed over the last 15 years for the computerized interchange of data important to structural crystallography, and for submissions to crystallographic publications and databases. The need for the volume stems from the IUCr's adoption in 1990 of CIF as the standard for exchanging crystallographic data across the entire community, and for the submission of research articles to *Acta Crystallographica* and other journals. The data items described in this volume appear as standard tags in CIF. Each data item has an associated definition, with attributes that characterize its allowed values and relationships with other defined items. The definitions provide a comprehensive machine-readable description of the information associated with a structural study, from area-detector counts through to the derived molecular geometry, and allow a full annotation suitable for publication as a primary research article.

The volume is arranged in five parts. Part 1 introduces the need and rationale for a standard interchange format, and the IUCr's commissioning and development of CIF in response to that need. Part 2 presents the formal specifications of the file formats involved in the CIF standard, and the related crystallographic binary file (CBF) and molecular information file (MIF) standards. Details are also given of the dictionary definition language (DDL) used to specify the names of data items and their attributes in a machine-readable way. The definitions and attributes of data items in different areas of crystallography are listed in separate dictionary files. Part 3 describes the structure and content of each of the public data dictionaries currently available. It provides commentaries on the dictionaries and guidance on best practice in their use for information interchange in several fields of crystallography. The dictionaries themselves are structured ASCII files that may be used directly by software parsers and validators. Part 4 presents the contents of the dictionaries in a text format that is easier for humans to read. Part 5 describes software applications and libraries that use CIF and related interchange standards.

The data definitions in this volume can be read and used from the printed page, but their principal use is in a computer-software environment. The volume therefore describes the large number of computer programs and libraries contributed by the crystallographic community for handling data files in the CIF format. Many of these programs use the dictionary files directly to validate and exchange data items. Software by its very nature is under continual development, and individual software implementations become obsolete with disturbing rapidity. Nevertheless, we have felt it important and useful to devote considerable space to the software aspects of CIF data exchange. The programs described in the volume illustrate various basic considerations and approaches to data exchange and provide a rich gallery of tools suitable for different applications. The volume also includes a CD-ROM containing many of these software packages, as well as machine-readable versions of the data dictionaries themselves, and many links to related web resources.

This volume therefore contains material that is relevant to both crystallographers and information-technology specialists. The data dictionaries that form the core of the volume provide the knowledge base for automated validation of data submitted for publication and for deposition in databases (tasks in which most structure-determination scientists are involved). Part 5 will be of particular interest to the non-specialist user of CIF, in that it contains practical advice on stand-alone application programs, the publication of crystal structure reports in primary research journals and the deposition of biological macromolecular structures in the Protein Data Bank. Software authors will benefit from the discussions on designing and implementing CIF-aware programs and from the detailed descriptions of a number of available libraries.

The data-representation and knowledge-management techniques used routinely in crystallography today were developed well in advance of recent approaches to information dissemination for the World Wide Web through open protocols. Nevertheless, CIF processes fit well with the enormous contemporary effort to build the so-called semantic web, in which information and knowledge are transported and interpreted by computer. The success of such knowledge management depends on well structured data vocabularies, data models and data structures appropriate for different subject areas. The term used to denote such semantically rich formulations is 'ontology'. We believe that the formal description of crystallographic data and information presented in this volume is an exemplary ontology in this sense, and will be a useful model for workers in related fields.

The volume has taken several years to complete. It covers subject matter that is relatively new and still changing, and indeed much of the material has been undergoing revision even as it was being compiled. Thanks to the determined effort by all involved in its production, this volume provides a comprehensive and up-to-date compendium of the current state of crystallographic data definition. This first edition also serves the role of providing the historical and scientific context for future developments in a rapidly evolving field. It is likely that new editions will be needed before long to stay abreast of such developments.

Since it began, the CIF project has involved the participation and collaboration of many colleagues across the breadth of crystallographic practice. In the production of this volume, we wish first to thank all the authors of the individual chapters for their major contributions. Each chapter was independently and anonymously peer-reviewed, and the reviewers' comments were forwarded to the authors for response. This has added to the time and effort involved on everyone's part, but we believe that the improved presentation and exposition of the detailed and often complex material will be of significant benefit to readers and users. We are grateful to the reviewers for their dedication and enthusiasm during this process. We particularly appreciate the help of many colleagues who have contributed to the CIF project generally and who have made valuable contributions to this volume. They are recognized as Associate Editors in the list of contributors. In addition to the individuals mentioned by name in individual chapters, we thank Doug du Boulay, Eldon Ulrich,

Frank Allen, Bill Clegg, Mark Spackman, Victor Streltsov, Bob Sweet, Howard Flack, Peter Murray-Rust and John Bollinger for thought-provoking discussions and contributions. We thank Philip Coppens for initially supporting the proposal for this volume, Theo Hahn and Hartmut Fuess, the previous and current Chairmen of the Commission on *International Tables for Crystallography*, and the other members of the Commission who welcomed this volume to the series. Thanks also go to Timo Vaalsta of the University of Western Australia for technical help with the CD-ROM. Finally, we are grateful to Nicola Ashcroft of the IUCr editorial office for her careful and insightful technical editing of this work.

1. HISTORICAL INTRODUCTION

BY S. R. HALL AND B. MCMAHON

1.1. Genesis of the Crystallographic Information File 2
1.1.1. Prologue 2
1.1.2. Past approaches to data exchange 2
1.1.3. Card-image formats 3
1.1.4. The Standard Crystallographic File Structure (SCFS) 3
1.1.5. The impact of networking on crystallography 4
1.1.6. The Working Party on Crystallographic Information (WPCI) 5
1.1.7. The Crystallographic Information File 6
1.1.8. Diversification: the Molecular Information File and dictionary definition language 7
1.1.9. The macromolecular Crystallographic Information File 7
1.1.10. The Crystallographic Binary File 8
1.1.11. Other extension dictionaries 8
1.1.12. The broader context: CIF and XML 9
References 9

1.1. Genesis of the Crystallographic Information File

BY S. R. HALL AND B. MCMAHON

1.1.1. Prologue

Progress in science depends crucially on the ability to find and share theories, observations and the results of experiments. The efficient exchange of information within and across scientific disciplines is therefore of fundamental importance. The rapid growth in the use of computers and networks over the past half century and in the use of the World Wide Web over the past decade have brought remarkable improvements in communications among scientists. Such communications are most effective when there is a common language. The English language has become the standard means of expression of ideas and theories; increasingly, the exchange of data requires the computer equivalent of such a *lingua franca*. It was with considerable foresight that the International Union of Crystallography (IUCr) in 1990 adopted a data-handling approach based on universal file concepts. At that time this was considered to be a radical idea. The approach adopted by the IUCr is known as the Crystallographic Information File (CIF). This volume of *International Tables for Crystallography* describes the CIF approach, the associated definition of CIF data items within dictionaries, and handling procedures, applications and software.

In this opening chapter, we give a historical perspective on the reasons why the CIF approach was adopted and how, over the past decade, CIF applications have evolved.

CIF is the most fully developed and mature of the various universal file approaches available today. It combines flexibility and simplicity of expression with a lean syntax. It has an unsurpassed ability to express 'hard' scientific data unambiguously using extensive dictionaries (ontologies) of relevant terms. It has proved to be remarkably well suited to the publication and archiving of small-unit-cell crystallographic structures. What was a radical idea in 1990 has today become the dominant mode of expression of scientific data in this domain.

The CIF data model provided the key to the internal restructuring of data managed by the Protein Data Bank in its transition from an archive to a database. The CIF approach is being tested in an increasing number of domains. In some cases, it may well become as successful as it has been for small-molecule crystallography. In other cases, the syntax will be unsuitable, but yet the conceptual discipline of agreed ontologies will still be required. Here, the experience of developing the CIF dictionaries may be carried across into different file formats and modes of expression.

Nowadays, informatics is a rapidly evolving field, in which everything is obsolete almost as soon as it is created. Yet there is a responsibility on today's scientists to preserve data and pass them on to the next generation. CIF was developed not only as a data-exchange mechanism, but also as an archival format, and considerable care has been taken over the past decade and more to keep it a stable and smoothly evolving approach. Some points of detail have been modified or superseded in practice. Other changes will necessarily occur as the approach evolves to meet the changing demands of an evolving science. Readers should therefore be aware of the need to consult the IUCr website (http://www.iucr.org/iucr-top/cif) for the latest versions of, or successors to, the data dictionaries and the software packages described in this volume. However, the basic concepts have already been shown to be remarkably effective and durable. This volume should therefore provide an invaluable reference for those working with CIF and related universal file approaches.

The success of any data-exchange approach depends on its efficiency and flexibility. It must cope with the increasing volume and complexity of data generated by the computing 'information explosion'. This growth challenges conventional criteria for measuring exchange and storage efficiency based on high data-compression factors. Today's fast, cheap magnetic and chip technologies make bulk volume a secondary consideration compared with extensibility and portability of data-management processes. Most importantly, improvements in computing technology continue to generate new approaches to harnessing semantic information contained within data collections and to promoting new strategies for knowledge management.

The basis for an efficient information-exchange process is mutually agreed rules for the supplier and the receiver, *i.e.* the establishment of an exchange protocol. This protocol needs to be established at several levels. At the first level, there must be predetermined ways that data (*i.e.* numbers, characters or text) are arranged in the storage medium. These are the organizational rules that define the syntax or the format of data. There must also be a clear understanding of the meaning of individual data items so that they can be correctly identified, accessed and reused by others. At an even higher level, a protocol may also provide rules for expressing the relationships between the data, as this can lead to automatic processes for validating and applying the data values.

These higher levels provide the *semantic* knowledge needed for the rigorous identification and validation of transmitted and stored data. One may consider the analogy of reading this paragraph in English. To do this we must first be able to recognize the individual words, comprehend their meaning (if necessary with the use of an English dictionary) and understand all of this within the context of sentence construction. As with data, the arrangement of component words is based on a predetermined grammatical syntax, and their individual meaning (as defined in a dictionary or elsewhere), coupled with their contextual function (as nouns, verbs, adjectives *etc.*), leads to the full comprehension of a word sequence as semantic information.

1.1.2. Past approaches to data exchange

The crystallographic community, along with many other scientific disciplines, has long adhered to the philosophy that experimental data and results should be routinely archived to facilitate long-term knowledge retention and access. An early approach to this, recommended by IUCr and other journals, was for authors to deposit data as hard copy (*i.e.* ink on paper) with the British Library Lending Division. Retaining good records is fundamental to reproducing

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

scientific results. However, the sheer volume of diffraction data needed to repeat a crystallographic study precludes these from publication, and has led in the past to relatively *ad hoc* procedures for depositing supplementary data in local or centralized archives. Typically in the past, only the crystal and structure model parameters were published in the refereed paper and the underpinning diffraction information had to be archived elsewhere. Because the archived data were usually stored as paper in various unregulated formats, considerable information about the experiment and structure-refinement parameters was never retained. Moreover, the archiving of supplementary data *via* postal services was very slow and labour-intensive; equally, the recovery of deposited data was difficult, with information supplied as either a photocopy of the original deposition or an image taken from a microfiche.

Prior to 1970, when less than 9000 structures were deposited with the Cambridge Crystallographic Data Centre, data sets were still small enough to make these deposition and retrieval approaches feasible, albeit tedious. Even so, records show that very few archived data sets were ever retrieved for later use. The rationale of data storage changed radically in the 1980s. The increasing role of computers, automatic diffractometers and phase-solving direct methods in crystallographic studies led to a rapid acceleration in the number and size of structures determined and published. This was the period when fast minicomputers became affordable for laboratories, and the consequent demand for the electronic storage and exchange of information grew exponentially. Typical data-archival practices changed from using paper to magnetic tapes, as these now became the least expensive and most efficient means of storing data.

1.1.3. Card-image formats

Although the interchange of scientific information depends implicitly on an agreed data format, it remains independent of whether the transmission medium is paper tape, punched card, magnetic tape, computer chip or the Internet. Crystallography has employed countless data-exchange approaches and formats over the past 60 years. Prior to the advent of computers, the standard approach involved the exchange of typed tables of coordinates and structure factors with descriptive headers. In the 1950s and 1960s, as computers became the dominant generators of data, the transfer of data between laboratories was still relatively uncommon. When it was necessary, the Hollerith card formats of commonly used programs, such as *ORFLS* (Busing *et al.*, 1962) and *XRAY* (Stewart, 1963), usually sufficed. Even when magnetic tape drives became common and were standardized (mainly to the 1/2-inch 2400-foot reel), the 80-column 'card-image' formats of these programs remained the most popular data exchange and deposition approach.

As the storage and transporting of electronic data became easier and cheaper, structural information was increasingly deposited directly in databases such as the Cambridge Structural Database (CSD; Allen, 2002) and the Protein Data Bank (PDB; Bernstein *et al.*, 1977). The CSD and PDB simplified these depositions by using standard layouts such as the ASER, BCCAB and PDB formats. Both the PDB and CSD used, and indeed still use as a backup deposition mode, fixed formats with 80-character records and identifier codes. Examples of these format styles are shown in Figs. 1.1.3.1 and 1.1.3.2.

The card-image approach, involving a rigid preordained syntax, survived for more than two decades because it was simple, and the suite of data types used to describe crystal structures remained relatively static.

```
HEADER    PLANT SEED PROTEIN                      30-APR-81   1CRN      1CRND  1
COMPND    CRAMBIN                                                       1CRN   4
SOURCE    ABYSSINIAN CABBAGE (CRAMBE ABYSSINICA) SEED                   1CRN   5
AUTHOR    W.A.HENDRICKSON,M.M.TEETER                                    1CRN   6
REVDAT   5   16-APR-87 1CRND   1       HEADER                          1CRND  2
REVDAT   4   04-MAR-85 1CRNC   1       REMARK                          1CRNC  1
REVDAT   3   30-SEP-83 1CRNB   1       REVDAT                          1CRNB  1
REVDAT   2   03-DEC-81 1CRNA   1       SHEET                           1CRNB  2
REVDAT   1   28-JUL-81 1CRN    0                                       1CRNB  3
REMARK    1                                                             1CRN   7
REMARK    1 REFERENCE 1                                                 1CRNC  2
REMARK    1  AUTH   M.M.TEETER                                          1CRNC  3
REMARK    1  TITL   WATER STRUCTURE OF A HYDROPHOBIC PROTEIN AT ATOMIC  1CRNC  4
REMARK    1  TITL 2 RESOLUTION. PENTAGON RINGS OF WATER MOLECULES IN     1CRNC  5
REMARK    1  TITL 3 CRYSTALS OF CRAMBIN                                 1CRNC  6
REMARK    1  REF    PROC.NAT.ACAD.SCI.USA         V.  81  6014 1984     1CRNC  7
REMARK    1  REFN   ASTM PNASA6  US ISSN 0027-8424                 040   1CRNC  8
REMARK    1 REFERENCE 2                                                 1CRNC  9
REMARK    1  AUTH   W.A.HENDRICKSON,M.M.TEETER                          1CRN   9
REMARK    1  TITL   STRUCTURE OF THE HYDROPHOBIC PROTEIN CRAMBIN        1CRN  10
REMARK    1  TITL 2 DETERMINED DIRECTLY FROM THE ANOMALOUS SCATTERING    1CRN  11
REMARK    1  TITL 3 OF SULPHUR                                          1CRN  12
REMARK    1  REF    NATURE                        V.  290  107 1981     1CRN  13
REMARK    1  REFN   ASTM NATUAS  UK ISSN 0028-0836                 006   1CRN  14
REMARK    1 REFERENCE 3                                                 1CRNC 10
REMARK    1  AUTH   M.M.TEETER,W.A.HENDRICKSON                          1CRN  16
REMARK    1  TITL   HIGHLY ORDERED CRYSTALS OF THE PLANT SEED PROTEIN    1CRN  17
REMARK    1  TITL 2 CRAMBIN                                             1CRN  18
REMARK    1  REF    J.MOL.BIOL.                   V.  127  219 1979     1CRN  19
REMARK    1  REFN   ASTM JMOBAK  UK ISSN 0022-2836                 070   1CRN  20
SEQRES    1     46  THR THR CYS CYS PRO SER ILE VAL ALA ARG SER ASN PHE  1CRN  51
SEQRES    2     46  ASN VAL CYS ARG LEU PRO GLY THR PRO GLU ALA ILE CYS  1CRN  52
SEQRES    3     46  ALA THR TYR THR GLY CYS ILE ILE ILE PRO GLY ALA THR  1CRN  53
SEQRES    4     46  CYS PRO GLY ASP TYR ALA ASN                         1CRN  54
HELIX    1  H1 ILE      7  PRO     19  1 3/10 CONFORMATION RES 17,19    1CRN  55
HELIX    2  H2 GLU     23  THR     30  1 DISTORTED 3/10 AT RES 30       1CRN  56
SHEET    1  S1 2 THR     1  CYS     4  0                                1CRNA  4
SHEET    2  S1 2 CYS    32  ILE    35 -1                                1CRN  58
TURN     1  T1 PRO    41  TYR    44                                     1CRN  59
SSBOND   1 CYS      3    CYS     40                                     1CRN  60
SSBOND   2 CYS      4    CYS     32                                     1CRN  61
SSBOND   3 CYS     16    CYS     26                                     1CRN  62
CRYST1   40.960   18.650   22.520  90.00  90.77  90.00 P 21          2  1CRN  63
ATOM      1  N   THR     1      17.047  14.099   3.625  1.00 13.79      1CRN  70
ATOM      2  CA  THR     1      16.967  12.784   4.338  1.00 10.80      1CRN  71
ATOM      3  C   THR     1      15.685  12.755   5.133  1.00  9.19      1CRN  72
ATOM      4  O   THR     1      15.268  13.825   5.594  1.00  9.85      1CRN  73
ATOM      5  CB  THR     1      18.170  12.703   5.337  1.00 13.02      1CRN  74
ATOM      6  OG1 THR     1      19.334  12.829   4.463  1.00 15.06      1CRN  75
ATOM      7  CG2 THR     1      18.150  11.546   6.304  1.00 14.23      1CRN  76
ATOM      8  N   THR     2      15.115  11.555   5.265  1.00  7.81      1CRN  77
ATOM      9  CA  THR     2      13.856  11.469   6.066  1.00  8.31      1CRN  78
ATOM     10  C   THR     2      14.164  10.785   7.379  1.00  5.80      1CRN  79
CONECT   20   19  282                                                   1CRN 398
CONECT   26   25  229                                                   1CRN 399
CONECT  116  115  188                                                   1CRN 400
CONECT  188  116  187                                                   1CRN 401
CONECT  229   26  228                                                   1CRN 402
CONECT  282   20  281                                                   1CRN 403
END                                                                     1CRN 405
```

Fig. 1.1.3.1. An abbreviated example of a PDB format file.

1.1.4. The Standard Crystallographic File Structure (SCFS)

By the 1980s, the many different fixed formats used to exchange data electronically had become a significant complication for journals and databases. Because of this, the IUCr Commissions for Crystallographic Data and Computing formed a joint Working Party which was asked to recommend a standard format for the exchange and retention of crystallographic data. They proposed a partially fixed format in which key words on each line identified blocks of data containing items in a specific order. This format was the Standard Crystallographic File Structure (Brown, 1988). An example of an SCFS file is shown in Fig. 1.1.4.1.

The effectiveness of the SCFS format approach was curtailed because its release coincided with the arrival of powerful minicomputers, such as the VAX780, in crystallographic laboratories. This led to a period of enormous change in crystallographic computing, in which new data types and file formats proliferated. It was also a time when automatic diffractometers became standard equipment in laboratories and the development of new crystallographic software packages flourished. The fixed-format design of the SCFS was unable to adapt easily to these continually changing data requirements, and this eventually led to a proliferation of SCFS versions.

```
?BAWGEL
#ADATE 820705
#COMPND bis(Benzene)-chromium bromide
#FORMUL C12 H12 Cr1 1+,Br1 1-
#AUTHOR A.L.Spek,A.J.M.Duisenberg
#JRNL 189,10,1531,1981
#CREF msdb 14.74.003 nbsid 532193 batch 53 cdvol 6
#CLASS 1/74
#SYSCAT sys O cat 3
#CONN El= Cr 2 Br 14 V= 1 2 Ch= + 2 Ch= - 14
Res= Plot= 1 B= 5 1-3 1-4 3-6 4-7 5-8 5-9 6-10 7-10 8-11 9-12 11-13 12-13
 B= 9 1-2-5
Res= Plot= 1 14
#DIAGRAM
   469   151   374   202   469   248   554   103   272   248   556   299
   640   152   186   296   272   151   640   250   101   247   185   100
   100   149   100    50     0     0     0     0     0     0     0     0
#CELL a 9.753(6) b 9.316(3) c 11.941(8) z 4 cent 1 sg Fmmm
#SYMM  x,y,z
x,1/2+y,1/2+z
1/2+x,y,1/2+z
1/2+x,1/2+y,z
-x,y,z
-x,1/2+y,1/2+z
1/2-x,y,1/2+z
1/2-x,1/2+y,z
x,-y,z
x,1/2-y,1/2+z
1/2+x,-y,1/2+z
1/2+x,1/2-y,z
-x,-y,z
-x,1/2-y,1/2+z
1/2-x,-y,1/2+z
1/2-x,1/2-y,z
#DENSITY dx 1.764
#UNIS int 3 sigcc 3
#RFACT R= 0.0540.
#RADIUS C  0.68 H  0.23 Br 1.21 Cr 1.35
#TOLER 0.40
#ATOM Cr1 0.0 0.0 0.0
Br1 0.0 0.0 0.50000
C1 0.06900 0.12800 0.13400
C2 0.13900 0.0 0.13400
H1 0.09300 0.20400 0.12500
H2 0.19800 0.0 0.13000
#BOND Cr1 C1 2.100
Cr1 C2 2.090
C1 C1* 1.340
C1 C2 1.370
C1 H1 0.760
C2 H2 0.580
#MDATE  901205
#END
```

Fig. 1.1.3.2. An example of a CSD BCCAB format file.

```
TITLE                                                                   00001
*p6122                  CIFIO  05-Mar87       p6122                     00002
                                                                        00003
SG NAME                                                                 00004
 LATT     NP                                                            00005
 SYST     HEXAGONAL                                                     00006
 BRAV     HEXAGONAL                                                     00007
 HALL     p_61_2_(0_0_-1)                                               00008
 HERM     p_61_2_2                                                      00009
*EOS                                                                    00010
                                                                        00011
SYMMETRY R11 2 3     T1      R21 2 3     T2      R31 2 3     T3         00012
 SYOP      1 0 0  .0000000     0 1 0  .0000000     0 0 1  .0000000  1    00013
 SYOP     -1 0 0  .0000000     0-1 0  .0000000     0 0 1  .5000000  2    00014
 SYOP      0-1 0  .0000000    -1 0 0  .0000000     0 0-1  .8333330  3    00015
 SYOP      0 1 0  .0000000     1 0 0  .0000000     0 0-1  .3333330  4    00016
 SYOP      1-1 0  .0000000     0-1 0  .0000000     0 0-1  .0000000  5    00017
 SYOP     -1 1 0  .0000000     0 1 0  .0000000     0 0-1  .5000000  6    00018
 SYOP      1 0 0  .0000000     1-1 0  .0000000     0 0-1  .1666670  7    00019
 SYOP     -1 0 0  .0000000    -1 1 0  .0000000     0 0-1  .6666670  8    00020
 SYOP      0-1 0  .0000000     1-1 0  .0000000     0 0 1  .3333330  9    00021
 SYOP      0 1 0  .0000000    -1 1 0  .0000000     0 0 1  .8333330 10    00022
 SYOP      1-1 0  .0000000     1 0 0  .0000000     0 0 1  .1666670 11    00023
 SYOP     -1 1 0  .0000000    -1 0 0  .0000000     0 0 1  .6666670 12    00024
*EOS                                                                    00025
                                                                        00026
FORMULA  EL  NUM                                                        00036
 FORL     s    .5000o    .5000c   1.0000                                00037
*EOS                                                                    00038
                                                                        00039
CONDITIONS                                                              00040
 CELLPAREX     .7107    566.00                                          00041
 INT PAREX     .7107    566.00      .147      .681        92            00042
 HKL PARE          0         2         0         4         0        12   00043
 EQIVPARE         92       525                                          00044
*EOS                                                                    00045
                                                                        00046
ATOMS    NAME    X U11   Y U22   Z U33   U U12   P U13     U23  MUL AT  DT 00052
 UALL              .03500                                               00053
 ATCO     s        .20140  .79860  .91667         1.00000          6  s   200054
 ATCE     s        .00040  .00040  .00000          .00000                00055
 UIJ      s        .04100  .04100  .01000  .02500 -.00400 -.00400        00056
 UIJE     s        .00800  .00800  .00700  .00700  .00500  .00500        00057
 ATCO     o        .50100  .50100  .66667         1.00000          6  o   200058
 ATCE     o        .00300  .00300  .00000          .00000                00059
 UIJ      o        .08900  .08900  .09000  .06300  .00900 -.00900        00060
 UIJE     o        .01800  .01800  .02000  .01900  .00800  .00800        00061
 ATCO     c 1      .49200  .09700  .03780         1.00000         12  c   200062
 ATCE     c 1      .00300  .00300  .00110          .00000                00063
 UIJ      c 1      .03170  .03170  .03170  .01585  .00000  .00000        00064
 UIJE     c 1      .00000  .00000  .00000  .00000  .00000  .00000        00065
*EOS                                                                    00066
                                                                        00067
CELL DIMENSIONS  A         B         C       ALPHA     BETA      GAMMA   Z 00068
 CELLPARE     8.5300    8.5300   20.3700   90.0000   90.0000  120.0000 12.000069
 ERRSPARE      .0100     .0100     .0100     .0100     .0100     .0100     00070
 VOL PARE   1283.571    3.0775               .5595                        00071
 PHYSPARE                      566.0000                                  00072
*EOS                                                                    00073
                                                                        00074
END                                                                     00081
```

Fig. 1.1.4.1. An abbreviated example of a Standard Crystallographic File Structure (SCFS) format file.

1.1.5. The impact of networking on crystallography

The growth in power of individual minicomputers inevitably helped the development of computational techniques in crystallography. Yet perhaps a more profound development was networking – the ability to exchange electronic data directly between computers. The laborious procedures for transferring information by manual keystroke or exchange of card decks and magnetic tapes were replaced by error-free programmatic procedures. Initially, data could flow easily between computers in the same laboratory; then colleagues could exchange data between scientific departments on the same campus; and before long experimental results, programs and general communications were flowing freely across national and international networks.

During the 1960s, networking was *ad hoc* and proprietary, and rarely extended effectively outside the laboratory. By the 1970s, however, a few standard networking protocols were becoming established. These included uucp, which promoted the growth of dial-up networking between university campuses, and TCP/IP, the transport protocol underlying the ARPANET, that would eventually give rise to the dominant Internet with which we are familiar today. The potential for improving the practice of crystallography through the ease of communications afforded by computer networks was very clear. However, the technology was still costly and required much effort and expertise to implement. Even towards the end of the decade, a meeting of protein crystallographers concluded (Freer & Stewart, 1979) that

> The possibility and usefulness of establishing a computer network for communication among crystallographic laboratories was discussed. The implications for rapid updating and the ease with which programs and data could be transfered among the groups was clearly recognized by all present; however, immediate implementation of a network was not deemed practical by a majority of the participants.

By the mid-1980s, the establishment of a global computer network was well under way. There was still some diversity of transmission protocols on an international scale: uucp, BITNET and X.25 Coloured Book protocols were still competing with TCP/IP, so that communication between different networks had to be managed through gateways. Nevertheless, there was sufficient standardization that it was feasible to communicate with colleagues world-wide by e-mail, to transfer files by ftp and to log in to remote computers by telnet. E-mail, in particular, allowed for the rapid transmission of ASCII text in an arbitrary format. In many respects, this established a goal for other exchange formats to achieve. The establishment of anonymous ftp sites permitted the free exchange of software and data to any user; no special privileges on the host computer were needed. Such availability of electronic information fitted particularly well with the scientific ethic of open exchange of information.

By the early 1990s, TCP/IP and the Internet dominated international networking. The practices of open exchange of information were developed through a number of initiatives. *Gopher* (Anklesaria *et al.*, 1993) provided a general mechanism to access material categorized and published from a computerized information store. *WAIS* (Kahle, 1991), a wide-area information server application designed to service queries conforming to the Z39.50 information retrieval protocol (ANSI/NISO, 1995), provided an effective distributed search engine. The rapid proliferation of new techniques for searching and retrieving information from the Internet was capped in the mid-1990s by the rapid growth in sites implementing hypertext servers (Berners-Lee, 1989). The World Wide Web had become a reality.

The increasing access to global network facilities during the 1980s led to a growing interest among crystallographers in submitting manuscripts to journals electronically, especially for small-molecule structure studies. The Australian delegation at the 1987 General Assembly of the XIVth IUCr Congress in Perth proposed that IUCr journals (specifically *Acta Crystallographica*) should be able to accept manuscripts submitted electronically. It was argued that this would reduce effort on the part of the authors and the journal office in preparation and transcription of manuscripts, and as a consequence reduce costs and transcription errors and simplify data-validation approaches. The acceptance of this General Assembly resolution led to the creation of a Working Party on Crystallographic Information (WPCI), which had as its mandate the investigation of possible approaches to enable the electronic submission of crystallographic research publications.

1.1.6. The Working Party on Crystallographic Information (WPCI)

The WPCI first convened at the 1988 ECM11 conference in Vienna. In the discussions leading up to this meeting, it was widely appreciated that electronic submissions to journals and databases involved data types (*e.g.* manuscript texts, graphical diagrams, the full suite of crystallographic data) that were beyond those accommodated within the SCFS format promoted by the IUCr Data and Computing Commissions. Consequently, it was suggested at the Vienna meeting that a general and extensible universal file approach, similar to the recently developed Self-defining Text Archive and Retrieval (STAR) File format (Hall, 1991; Hall & Spadaccini, 1994), might also be suitable for crystallographic data applications.

At this meeting, it was decided that a WPCI working group, led by Syd Hall, should investigate the development of a universal file protocol that would be suitable for crystallographic data needs. Other universal formats existed, such as ASN.1 (ISO, 2002), which was used for data communications, JCAMP-DX (McDonald & Wilks, 1988), which was used for archiving infrared spectra, and the Standard Molecular Data (SMD) format (Barnard, 1990), which was used for the global exchange of chemical structure data. These were considered relatively inefficient for expressing the repetitive data lists commonly used in crystallography. The working group eventually proposed a Crystallographic Information File (CIF) format which had a syntax similar to, but simpler than, the STAR File. Of particular importance because of the rapid changes taking place with data types, the CIF approach provided a very flexible and extensible file structure in which any type of text or numerical data could be arranged in any order. The typical data structure of a CIF is illustrated in Fig. 1.1.6.1, using the same data as presented in the PDB file of Fig. 1.1.3.1. Similarly, Fig. 1.1.6.2 shows the data in the BCCAB file of Fig. 1.1.3.2 in CIF format.

```
data_crambin
_entry.id                          1CRN

_audit.creation_date               1993-04-21
_audit.creation_method             'manual editing of PDB entry'
_audit.update_record
; 1993-04-21 Original PDB entry history recorded here for completeness.
        30-apr-81 deposition.
        28-jul-81 1crn    0
        03-dec-81 correct residue number on strand 1 of sheet s1.
        30-sep-83 insert revdat records
        04-mar-85 insert new publication as reference and renumber
        16-apr-87 change deposition date from 31-apr-81 to 30-apr-81.
;
loop_
    _struct.entry_id
    _struct.title
    1CRN  'Crambin from Abyssinian cabbage (Crambe abyssinica) seed'

loop_
    _citation.id
    _citation.year
    _citation.journal_abbrev
    _citation.journal_volume
    _citation.page_first
    _citation_journal_id_ASTM
    _citation_journal_id_ISSN
    _citation_title
    primary   1984  Biochemistry  23  6796
            ?      0006-2960
;   Raman spectroscopy of homologous plant toxins: crambin and alpha 1- and
    beta-purothionin secondary structures, disulfide conformation, and
    tyrosine environment
;
    1     1984  Proc.Nat.Acad.Sci.USA  81   6014
            pnasa6   0027-8424
;   Water structure of a hydrophobic protein at atomic resolution. Pentagon
    rings of water molecules in crystals of crambin
;
    2     1981  Nature                 280  107
            natuas   0028-0836
;   Structure of the hydrophobic protein crambin determined directly
    from the anomalous scattering of sulphur
:
loop_
    _citation_author.citation_id
    _citation_author.name
    primary   'Williams, R.W.'        primary   'Teeter, M.M.'
    1         'Teeter, M.M.'
    2         'Hendrickson, W.A.'     2         'Teeter, M.M.'

loop_
    _entity.id
    _entity.type
    _entity.details
    1     polymer       'Protein chain: *'
    2     non-polymer   'het group EOH'

loop_
    _entity_poly_seq.entity_id
    _entity_poly_seq.num
    _entity_poly_seq.mon_id
       1   1 THR    1   2 THR    1   3 CYS    1   4 CYS    1   5 PRO
       1   6 SER    1   7 ILE    1   8 VAL    1   9 ALA    1  10 ARG
       1  11 SER    1  12 ASN    1  13 PHE    1  14 ASN    1  15 VAL
#.........................................sequence data omitted for brevity

_cell.length_a                    40.960
_cell.length_b                    18.650
_cell.length_c                    22.520
_cell.angle_alpha                 90.00
_cell.angle_beta                  90.77
_cell.angle_gamma                 90.00
_symmetry.space_group_name_H-M 'P 1 21 1'

loop_
    _atom_type.symbol
    _atom_type.description
    _atom_type.number_in_cell
    C carbon 404  N nitrogen 112  O oxygen 128  S sulfur 12  H hydrogen ?

loop_
    _atom_site.label_seq_id
    _atom_site.type_symbol
    _atom_site.label_atom_id
    _atom_site.label_comp_id
    _atom_site.label_asym_id
    _atom_site.auth_seq_id
    _atom_site.label_alt_id
    _atom_site.Cartn_x
    _atom_site.Cartn_y
    _atom_site.Cartn_z
    _atom_site.occupancy
    _atom_site.B_iso_or_equiv
    _atom_site.label_entity_id
    _atom_site.id
      1  N  N    THR *  1  .  17.047  14.099  3.625  1.00 13.79  1   1
      1  C  CA   THR *  1  .  16.967  12.784  4.338  1.00 10.80  1   2
      1  C  C    THR *  1  .  15.685  12.755  5.133  1.00  9.19  1   3
      1  O  O    THR *  1  .  15.268  13.825  5.594  1.00  9.85  1   4
      1  C  CB   THR *  1  .  18.170  12.703  5.337  1.00 13.02  1   5
      1  O  OG1  THR *  1  .  19.334  12.829  4.463  1.00 15.06  1   6
      1  C  CG2  THR *  1  .  18.150  11.546  6.304  1.00 14.23  1   7
#.........................................atom-site data omitted for brevity
```

Fig. 1.1.6.1. Example 1 of a CIF (using the same data as shown in Fig. 1.1.3.1).

```
data_BAWGEL
_audit_creation_date              93-05-24
_audit_creation_method            manual_conversion_of_ccdc_file
_audit_update_record
;  82-07-05  CCDC entry created from journal data
                     A.L.Spek,A.J.M.Duisenberg (1981) 189,10,1531
   93-05-21  Received file from Owen Johnson, CCDC.
   93-05-24  Initial conversion of the file to CIF/MIF format.
;
_chemical_name_systematic         'bis(Benzene)-chromium bromide'
_chemical_formula_moiety          'C12 H12 Cr1 1+,Br1 1-'

_cell_length_a                    9.735(6)
_cell_length_b                    9.316(3)
_cell_length_c                    11.941(8)
_cell_angle_alpha                 90
_cell_angle_beta                  90
_cell_angle_gamma                 90
_cell_formula_units_Z             4

_symmetry_space_group_name_H-M    Fmmm

loop_
    _symmetry_equiv_pos_as_xyz
        x,y,z x,1/2+y,1/2+z 1/2+x,y,1/2+z 1/2+x,1/2+y,z -x,y,z -x,1/2+y,1/2+z
        1/2-x,y,1/2+z 1/2-x,1/2+y,z  x,-y,z x,1/2-y,1/2+z 1/2+x,-y,1/2+z
        1/2+x,1/2-y,z -x,-y,z -x,1/2-y,1/2+z 1/2-x,-y,1/2+z 1/2-x,1/2-y,z

loop_
    _atom_type_symbol
    _atom_type_radius_bond
                           C 0.68   H 0.23   Br 1.21   Cr 1.35

_exptl_crystal_density_meas       1.764
_refine_ls_R_factor_obs           0.0540
_reflns_observed_criterion        3sigma(I)

loop_
    _atom_site_label
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
                      Cr1 0.0     0.0     0.0
                      Br1 0.0     0.0     0.50000
                      C1  0.06900 0.12800 0.13400
                      C2  0.13900 0.0     0.13400
                      H1  0.09300 0.20400 0.12500
                      H2  0.19800 0.0     0.13000
loop_
    _geom_bond_atom_site_label_1
    _geom_bond_atom_site_label_2
    _geom_bond_distance
    _geom_bond_site_symmetry_1
    _geom_bond_site_symmetry_2

                         Cr1 C1 2.100  .   .
                         Cr1 C2 2.090  .   .
                         C1  C1 1.340  .   5_555
                         C1  C2 1.370  .   .
                         C1  H1 0.760  .   .
                         C2  H2 0.580  .   .
```

Fig. 1.1.6.2. Example 2 of a CIF (using the same data as shown in Fig.1.1.3.2).

1.1.7. The Crystallographic Information File

As outlined in Section 1.1.6, the working group commissioned by the WPCI set out to establish an exchange protocol suitable for submitting crystallographic data to journals and databases, and this resulted in the development of the CIF syntax. At the same time, the group was also asked to form a list of those data items considered to be essential in a manuscript submitted to *Acta Crystallographica*. The data items originally recommended are listed in Fig. 1.1.7.1.

The syntax of a CIF (a detailed description is given in Chapter 2.2) was intentionally a simple subset of the STAR File syntax (see Chapter 2.1 for details). This simplification was considered important for its easy implementation in existing crystallographic software packages – clearly a primary goal for any format that was to be widely available for submitting data to journals and databases.

A compilation of data names referring to specific quantities or concepts in a crystal-structure determination was drawn up. This compilation included the items already identified as necessary for publication and many more besides. As a list of standard tags intended for unambiguous use, the collection was known from the outset as a *dictionary* of data names.

```
# The following CIF data names encompass the IUCr Journals Commission
# requirements for the reporting of a small-molecule structure in Acta
# Crystallographica, Section C

_chemical_compound_source
_chemical_formula_sum
_chemical_formula_moiety
_chemical_formula_weight
_symmetry_cell_setting
_symmetry_space_group_name_H-M
_cell_length_a
_cell_length_b
_cell_length_c
_cell_angle_alpha
_cell_angle_beta
_cell_angle_gamma
_cell_volume
_cell_formula_units_Z
_exptl_crystal_density_diffrn
_exptl_crystal_density_meas
_exptl_crystal_density_method
_diffrn_radiation_type
_diffrn_radiation_wavelength
_cell_measurement_reflns_used
_cell_measurement_theta_min
_cell_measurement_theta_max
_exptl_absorpt_coefficient_mu
_cell_measurement_temperature
_exptl_crystal_description
_exptl_crystal_colour
_exptl_crystal_size_max
_exptl_crystal_size_mid
_exptl_crystal_size_min
_exptl_crystal_size_rad
_diffrn_measurement_device
_diffrn_measurement_method
_exptl_absorpt_correction_type
_exptl_absorpt_correction_T_min
_exptl_absorpt_correction_T_max
_diffrn_reflns_number
_reflns_number_total
_reflns_number_observed
_reflns_observed_criterion
_diffrn_reflns_av_R_equivalents
_diffrn_reflns_theta_max
_diffrn_reflns_limit_h_min
_diffrn_reflns_limit_h_max
_diffrn_reflns_limit_k_min
_diffrn_reflns_limit_k_max
_diffrn_reflns_limit_l_min
_diffrn_reflns_limit_l_max
_diffrn_standards_number
_diffrn_standards_interval_count
_diffrn_standards_interval_time
_diffrn_standards_decay_%
_refine_ls_structure_factor_coef
_refine_ls_R_factor_obs
_refine_ls_wR_factor_obs
_refine_ls_goodness_of_fit_obs
_refine_ls_number_reflns
_refine_ls_number_parameters
_refine_ls_hydrogen_treatment
_refine_ls_weighting_scheme
_refine_ls_shift/esd_max
_refine_diff_density_max
_refine_diff_density_min
_refine_ls_extinction_method
_refine_ls_extinction_coef
_atom_type_scat_source
_refine_ls_abs_structure_details
_computing_data_collection
_computing_cell_refinement
_computing_data_reduction
_computing_structure_solution
_computing_structure_refinement
_computing_molecular_graphics
_computing_publication_material
_publ_section_experimental
```

Fig. 1.1.7.1. Initial set of data items considered to be essential in a structure report submitted to *Acta Crystallographica Section C*.

The WPCI proposed the CIF format as a standard exchange protocol at the open meetings of the IUCr Commissions on Crystallographic Data and Computing at the 1990 XVth IUCr Congress in Bordeaux. The proposal was accepted and the CIF format was subsequently adopted by the IUCr as the preferred format for data exchange (Hall *et al.*, 1991).

The administration of the CIF standard, including the approval of new data items, is the responsibility of the IUCr Committee for the Maintenance of the CIF Standard (COMCIFS). This committee plays a central role in the coordination of CIF activities, such as the creation of new dictionaries for defining crystallographic data items and the updating of data definitions in existing dictionaries. Chapter 3.1 describes relevant aspects of its role in the

commissioning and maintenance cycle. Information about COMCIFS activities and other CIF developments may be obtained from http://www.iucr.org/iucr-top/cif/.

1.1.8. Diversification: the Molecular Information File and dictionary definition language

While the primary thrust of these activities was the development of an exchange mechanism for crystal-structure reports, there was also interest in enriching the description of the chemical properties and behaviour of the compounds under study. Some work was therefore done to broaden the descriptions of bond order that were present in rudimentary form in the core CIF dictionary and to develop more detailed two-dimensional graphical representations of chemical molecules. The result of this work was the Molecular Information File (MIF), described in Chapter 2.4. As with CIF, the specific data items required for MIF were defined in a dictionary.

This work has extended the STAR File approach into chemistry. It is envisaged that later modules could describe spectroscopic data, reaction schemes and much more. Particular requirements of chemical structural databases are the need to query for generic structures, and the need to allow for the labelling and comparison of libraries of substructural components. Both requirements can be met by features of the STAR File, but they are features omitted from CIF. In practice, therefore, data files in MIF format cannot be readily accessed by most crystallographic applications, and the format is at present little used by crystallographers.

An important outcome of the work on MIF was the recognition that attributes of the data items needed for a particular application can be recorded using the same formalism as the data files themselves. This gave rise to the idea of a dictionary definition language (DDL), a set of tags for describing the names and attributes of data items. The dictionaries (the collections of data names for CIF and MIF applications) could then be constructed as STAR Files themselves, with the immediate result that software written to parse data files could equally easily parse the associated dictionaries. Now it became feasible to build into applications the ability to validate data by dynamically reading and interpreting the properties associated with a data tag in an accompanying dictionary.

The idea of a DDL was proposed by Tony Cook during early discussions on MIF and was adopted while the original CIF paper (Hall *et al.*, 1991) was in the press. The original core CIF dictionary was therefore produced with an early version of a DDL that was never fully documented (Fig. 1.1.8.1). Building on early experience with the core dictionary and the technical evolution of MIF, Hall & Cook (1995) worked through several revisions before publishing DDL version 1.4, the stable version described in Chapter 2.5 of this volume. Because of the circumstances in which it was developed, this dictionary definition language is able to accommodate both the flat-file quasi-relational structure of CIF and the more hierarchical multiple-looped data model of MIF.

1.1.9. The macromolecular Crystallographic Information File

The original goal of CIF was the creation of an archive format for the description and results of experiments in small-molecule and inorganic crystallography. The data names and their definitions were embodied in the *core* dictionary, so called because most of the terms in it were considered common to any crystallographic application. In 1990, the IUCr formed a working group, chaired by Paula Fitzgerald, to expand this dictionary to meet the additional requirements of macromolecular crystallography. The resulting expanded dictionary was to be known as the macromolecular CIF (mmCIF) dictionary.

```
##############################################################################
#
#                        DDL Data Name Descriptions
#                        --------------------------
#
# _compliance           The dictionary version in which the item is defined.
#
# _definition           The description of the item.
#
# _enumeration          A permissible value for an item. The value 'unknown'
#                       signals that the item can have any value.
#
# _enumeration_default  The default value for an item if it is not specified
#                       explicitly. 'unknown' means the default is not known.
#
# _enumeration_detail   The description of a permissible value for an item.
#                       Note that the code '.' normally signals a null
#                       or 'not applicable' condition.
#
# _enumeration_range    The range of values for a numerical item. The
#                       construction is 'min:max'. If 'max' is omitted then the
#                       item can have any value greater than or equal to 'min'.
#
# _esd                  Signals whether an estimated standard deviation is
#                       expected to be appended (enclosed within brackets)
#                       to a numerical item. May be 'yes' or 'no'.
#
# _esd_default          The default value for the esd of a numerical item
#                       if a value is not appended.
#
# _example              An example of the item.
#
# _example_detail       A description of the example.
#
# _list                 Signals whether an item is expected to occur in a looped
#                       list. Possible values: 'yes','no' or 'both'.
#
# _list_identifier      Identifies a data item that MUST appear in the list
#                       containing the currently defined data item.
#
# _name                 The data name of the item defined.
#
# _type                 The data type 'numb' or 'char' (latter includes 'text').
#
# _units_extension      The data-name extension code used to specify the units
#                       of a numerical item.
#
# _units_description    A description of the units.
#
# _units_conversion     The method of converting the item into a value based
#                       on the default units. Each conversion number is
#                       preceded by an operator code *, /, +, or - which
#                       indicates how the conversion number is applied.
#
# _update_history       A record of the changes to this file.
#
#-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof-eof
```

Fig. 1.1.8.1. Informal DDL used in the initial version of the core CIF dictionary (now superseded by DDL1.4).

The group's original short-term goal was to fulfil the IUCr mandate of defining data names for an adequate description of a macromolecular crystallographic experiment and its results. Longer-term goals were also determined: to provide sufficient data names so that the experimental section of a structure paper could be written automatically and to facilitate the development of tools so that computer programs could easily interface with CIF data files. A number of informal and formal meetings were held to describe the progress of this project and to solicit community feedback.

An important meeting took place at the University of York in April 1993. The attendees included the mmCIF working group, structural biologists and computer scientists. Vigorous discussion arose on whether the formal structure of the dictionary implemented in the then-current dictionary definition language (DDL1) could deal with the complexity of macromolecular data sets. There were criticisms that the data typing was not strong enough and that there were no formal links expressing relationships between data items. A working group was formed to address these issues, resulting in a second workshop in Tarrytown, New York, in October 1993. The discussions at this second meeting focused on the development of software tools and the requirements of an enhanced DDL. Such a DDL was proposed during a third workshop at the Free University of Brussels in October 1994. This new DDL (DDL2; Chapter 2.6) was designed by John Westbrook and sought to address the various problems identified at the preceding workshops, while retaining compatibility with existing CIF data files.

The macromolecular dictionary was built using DDL2 and was presented as a draft to the community at the American Crystallographic Association meeting in Montreal in July 1995. The draft was subsequently posted on a website and community comment solicited *via* an e-mail discussion list. This provoked lively discussions, leading to continuous correction and updating of the dictionary over an extended period of time. Software for parsing the dictionary and managing mmCIF data sets was developed and was also presented on the website.

In January 1997, the mmCIF dictionary was completed and submitted to COMCIFS for review. In June 1997, version 1.0 was approved by COMCIFS and released (Bourne *et al.*, 1997; Fitzgerald *et al.*, 1996). A workshop was held at Rutgers University in October 1997, at which tutorials were presented to demonstrate the use of the various tools that had been developed.

More than 100 new definitions have been incorporated in version 2 of the mmCIF dictionary, presented in Chapter 4.5.

1.1.10. The Crystallographic Binary File

In 1995, Andy Hammersley approached the IUCr with a proposal to develop an exchange and archival mechanism for image data using the CIF formalism. There was an increasing need to record and exchange two-dimensional images from a growing collection of area detectors and image plates from several manufacturers, each using different and proprietary data-storage formats. The project was encouraged by the IUCr and was developed under a variety of working names. In October 1997, a workshop at Brookhaven National Laboratory, New York, was convened to discuss progress and to coordinate the development of software suitable for this new format.

At the workshop, it became apparent that there was broad consensus to adopt and further develop the working set of data names that had been devised by the working group on this project; it was further decided that the relationships between these data names were best handled by the DDL2 formalism used for mmCIF. While image plates were not the sole preserve of macromolecular crystallography, it was felt that there would be maximum synergy with that community, and that the proposed imgCIF dictionary was most naturally viewed as an extension or companion to the mmCIF dictionary. The adoption of DDL2 would not preclude the generation of DDL1 analogue files if they were found to be necessary in certain applications, but to date such a need has not arisen.

However, on one point the workshop was insistent. For efficient handling of large volumes of image data on the necessary timescales within a large synchrotron research facility, the raw data must be in binary format. It was argued that although ASCII encoding could preserve the information content of a data stream in a fully CIF-compliant format, the consequent overheads in increased file size and data-processing time were unacceptable in environments with a very high throughput of such data sets, even given the high performance of modern computers. Consequently, the Crystallographic Binary File (CBF) was born as an extension to CIF (Chapter 2.3). The CBF may contain binary data, and therefore cannot be considered a CIF (Chapter 2.2). However, except for the representation of the image data, the file retains all the other features of CIF. Information about the experimental apparatus, duration, environmental conditions and operating parameters, together with descriptions of the pixel characteristics of individual frames, are all provided in ASCII character fields tagged by data names that are themselves fully defined in a DDL2-based dictionary (Chapter 3.7).

At first sight, the distinction between CIF and the crystallographic binary file may therefore seem trivial. However, in allowing binary data, the CBF requires greater care in defining the structure and packing by octets of the data, and loses some of the portability of CIF. It also precludes the use of many simple text-based file-manipulation tools. On the other hand, the importing of a stable and well developed tagged information format means that developers do not need to write novel parsers and compatibility with CIFs is easily attained. Indeed, the original idea of a fully compliant image CIF has been retained. By ASCII-encoding the image data in a crystallographic binary file, a fully compliant CIF, known as imgCIF, may be simply generated. One could consider an imgCIF as an archival version and a CBF as a working version of the same information set.

1.1.11. Other extension dictionaries

Since the introduction of the major CIF dictionaries, several other compendia of data names suitable for describing different applications or disciplines within crystallography have been developed. Four of these are described in the current volume.

The powder CIF dictionary (pdCIF; Chapter 3.3) is a supplement to the core dictionary addressing the needs of powder diffractionists. The structural model derived from powder work is familiar to single-crystal small-molecule or inorganic scientists. However, the powder CIF effort had the additional goals of documenting and archiving experimental data. It was always intended that powder CIF be used for communication of completed studies and for data exchange between laboratories. This is frequently done at shared diffraction facilities such as neutron and synchrotron sources. The powder dictionary was written with data from conventional X-ray diffractometers and from synchrotron, continuous-wavelength neutron, time-of-flight neutron and energy-dispersive X-ray instruments in mind.

The modulated-structures dictionary (msCIF; Chapter 3.4) is also considered as a supplement to the core dictionary and is designed to permit the description of incommensurately modulated crystal structures. The project was sponsored by the IUCr Commission on Aperiodic Crystals and was developed in parallel with a standard for the reporting of such structures in the literature (Chapuis *et al.*, 1997).

A small dictionary of terms for reporting accurate electron densities in crystals (rhoCIF; Chapter 3.5) has recently been published as a further supplement to the core dictionary. It has been developed under the sponsorship of the IUCr Commission on Charge, Spin and Momentum Densities.

The symmetry dictionary (symCIF; Chapter 3.8) was developed under the direct sponsorship of COMCIFS with the objective of producing a rigorous set of definitions suitable for the description of crystallographic symmetry. Following its publication, several data names from the symCIF dictionary were incorporated in the latest version of the core dictionary to replace the original informal definitions relating to symmetry. The symCIF dictionary contains most of the data names that would be needed to tabulate space-group-symmetry relationships in the manner of *International Tables for Crystallography* Volume A (2002). It is intended to expand the dictionary to include group–subgroup relations in a later version.

Other dictionaries are also under development, often under the supervision of one of the Commissions of the International Union of Crystallography.

Mention should also be made of the use of STAR Files by the BioMagResBank group at the University of Wisconsin to record NMR structures. This work (Ulrich *et al.*, 1998) endeavours to be complementary to the mmCIF descriptions of structures in the Protein Data Bank.

1.1.12. The broader context: CIF and XML

In the light of more recent data-exchange developments, it will be surprising to newcomers to CIF that more use is not made in crystallography of the extensible markup language XML (W3C, 2001). However, the development of CIF predates XML, and the CIF format can be easily translated to and from suitable XML representations. Most current crystallographic software imports and exports data in CIF format and the use of XML only becomes important in applications that cross the boundary of crystallography and involve interoperability with other scientific domains.

At one time, the antecedent of XML, standard generalized markup language (SGML; ISO, 1986), was considered as a candidate for a crystallographic exchange mechanism. SGML is a highly flexible and extensible system for specifying markup languages, but is extremely general. In the late 1980s, successful SGML implementations stretched the capacity of affordable computers and little accompanying software was available. SGML was at that time a suitable data and document tagging mechanism for large-scale publishers, but was far from appropriate for smaller-scale applications. XML was introduced during the 1990s as a specific SGML markup with a concrete syntax and simplifications that resulted in a lightweight, manageable language, for which robust parsers, editors and other programs could be written and implemented on desktop computers. The consequence has been a very rapid adoption of XML across many disciplines. Parallels may be drawn with the decision to implement CIF as a subset of the more general STAR File.

XML provides the ability to mark up a document or data set with embedded tags. Such tags may indicate a particular typographic representation. More usefully, however, they can reflect the nature or purpose of the information to which they refer. The design of such useful and well structured content tagging (in XML and other formalisms) is referred to in terms of constructing a subject or domain *ontology*. This term is rather poorly defined, but broadly covers the construction for a specific topic area of a controlled vocabulary of terms, the elaboration of relationships between those terms, and rules or constraints governing the use of the terms.

In XML and SGML, document-type definitions (DTDs) and schemas exist as external specifications of the markup tags permitted in a document, their relationships and any optional attributes they might possess. If carefully designed, these have the potential to act as ontologies. A simplification that XML offers over SGML is the ability to construct documents that do not need to conform to a particular schema. This makes it rather easier to develop software for generating and transmitting XML files. However, if diverse applications are to make use of the information content in an XML file, there must be some general way to exchange information about the meaning of the embedded markup, and in practice DTDs or schemas are essential for interoperability between software applications from different sources.

Recent initiatives in chemistry, under the aegis of the International Union of Pure and Applied Chemistry (IUPAC), suggest an active interest in the development of a machine-parsable ontology for chemistry. Building on an existing XML representation of chemical information known as Chemical Markup Language (CML; Murray-Rust & Rzepa, 1999, 2001), IUPAC project groups are mapping out areas of the science for which suitable DTDs and schemas may be constructed that tag relevant chemical content.

In this context, the CIF dictionaries described and annotated at length in this volume provide a sound basis for a machine-parsable ontology for crystallographic data. Given the orderly classification and relationships between tags in a CIF data set, format transformations using XML tags are not at all difficult to achieve. Chapters 5.3 and 5.5 describe tools for converting between mmCIF and XML formats for interchange within the biological structure community.

In the future, we expect to see the growth of XML environments where the full content of the CIF dictionaries is imported for the purpose of tagging crystallographic data embedded in more general documents and data sets. There have already been examples of ontology development using CIF dictionaries; for example, the Object Management Group has developed software classes for middleware in biological computing applications that are modelled on the mmCIF dictionary (Greer, 2000). The use of interactive STAR ontologies is described by Spadaccini *et al.* (2000).

It is possible that as interdisciplinary knowledge-management software systems develop, the exchange of crystallographic data will occur through XML files or other formats. Neverthless, CIF will remain an efficient mechanism for developing detailed data models within crystallography. We can confidently say that, whatever the actual transport format, the intellectual content of the dictionaries in this volume and their subsequent extensions and revisions will continue to underpin the definition and exchange of crystallographic data.

References

Allen, F. H. (2002). *The Cambridge Structural Database: a quarter of a million crystal structures and rising. Acta Cryst.* B**58**, 380–388.

Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D. & Alberti, B. (1993). *The Internet gopher protocol (a distributed document search and retrieval protocol).* RFC 1436. Network Working Group. http://www.ietf.org/rfc/rfc1436.txt.

ANSI/NISO (1995). *Information retrieval (Z39.50): Application service definition and protocol specification.* Z39.50-1995. http://lcweb.loc.gov/z3950/agency/1995doce.html.

Barnard, J. M. (1990). *Draft specification for revised version of the Standard Molecular Data (SMD) format. J. Chem. Inf. Comput. Sci.* **30**, 81–96.

Berners-Lee, T. (1989). *Information management: a proposal.* Internal report. Geneva: CERN. http://www.w3.org/History/1989/proposal-msw.html.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F. Jr, Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *The Protein Data Bank: a computer-based archival file for macromolecular structures. J. Mol. Biol.* **112**, 535–542.

Bourne, P., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. & Fitzgerald, P. M. D. (1997). *Macromolecular Crystallographic Information File. Methods Enzymol.* **277**, 571–590.

Brown, I. D. (1988). *Standard Crystallographic File Structure-87. Acta Cryst.* A**44**, 232.

Busing, W. R., Martin, K. O. & Levy, H. A. (1962). *ORFLS.* Report ORNL-TM-305. Oak Ridge National Laboratory, Tennessee, USA.

Chapuis, G., Farkas-Jahnke, M., Pérez-Mato, J. M., Senechal, M., Steurer, W., Janot, C., Pandey, D. & Yamamoto, A. (1997). *Checklist for the description of incommensurate modulated crystal structures. Report of the International Union of Crystallography Commission on Aperiodic Crystals. Acta Cryst.* A**53**, 95–100.

Fitzgerald, P. M. D., Berman, H., Bourne, P., McMahon, B., Watenpaugh, K. & Westbrook, J. (1996). *The mmCIF dictionary: community review and final approval. Acta Cryst.* A**52** (Suppl.), C575.

Freer, S. T. & Stewart, J. (1979). *Computer programming for protein crystallographic applications, University of California at San Diego, 28-29 November 1978. J. Appl. Cryst.* **12**, 426–427.

Greer, D. S. (2000). *Macromolecular structure RFP response, revised submission.* http://openmms.sdsc.edu/OpenMMS-1.5.1_Std/openmms/docs/specs/lifesci_00-11-01.pdf.

Hall, S. R. (1991). *The STAR file: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

International Tables for Crystallography (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn. Dordrecht: Kluwer Academic Publishers.

ISO (1986). ISO 8879. *Information processing – Text and office systems – Standard Generalized Markup Language (SGML).* Geneva: International Organization for Standardization.

ISO (2002). ISO/IEC 8824-1. *Abstract Syntax Notation One (ASN.1). Specification of basic notation.* Geneva: International Organization for Standardization.

Kahle, B. (1991). *An information system for corporate users: wide area information servers.* Thinking Machines technical report TMC-199. See also *Online*, **15**, 56–60.

McDonald, R. S. & Wilks, P. A. (1988). *JCAMP-DX: a standard form for exchange of infrared spectra in computer readable form. Appl. Spectrosc.* **42**, 151–162.

Murray-Rust, P. & Rzepa, H. S. (1999). *Chemical markup language and XML. Part I. Basic principles. J. Chem. Inf. Comput. Sci.* **39**, 928–942.

Murray-Rust, P. & Rzepa, H. S. (2001). *Chemical markup, XML and the world-wide web. Part II: Information objects and the CMLDOM. J. Chem. Inf. Comput. Sci.* **41**, 1113–1123.

Spadaccini, N., Hall, S. R. & Castleden, I. R. (2000). *Relational expressions in STAR File dictionaries*. *J. Chem. Inf. Comput. Sci.* **40**, 1289–1301.

Stewart, J. (1963). *XRAY63 Crystal Structure Calculations System.* Report TR-64-6 (NSG-398). Computer Science Center, University of Maryland, USA, and Research Computer Center, University of Washington, USA.

Ulrich, E. L. *et al.* (1998). XVIIth Intl Conf. Magn. Res. Biol. Systems. Tokyo, Japan.

W3C (2001). *Extensible Markup Language (XML).* http://www.w3c.org/XML/.

2. CONCEPTS AND SPECIFICATIONS

BY F. H. ALLEN, J. M. BARNARD, H. M. BERMAN, H. J. BERNSTEIN, I. D. BROWN, A. P. F. COOK, S. R. HALL, A. P. HAMMERSLEY, B. MCMAHON, N. SPADACCINI AND J. D. WESTBROOK

2.1. Specification of the STAR File (S. R. HALL AND N. SPADACCINI) 13
2.1.1. Introduction 13
2.1.2. Universal data language concepts 13
2.1.3. The syntax of the STAR File 14
Appendix 2.1.1. Backus–Naur form of the STAR syntax and grammar 16
References 19
2.2. Specification of the Crystallographic Information File (CIF) (S. R. HALL AND J. D. WESTBROOK) 20
2.2.1. Introduction 20
2.2.2. Terminology 20
2.2.3. The syntax of a CIF 21
2.2.4. Portability and archival issues 22
2.2.5. Common semantic features 23
2.2.6. CIF metadata and dictionary compliance 25
2.2.7. Formal specification of the Crystallographic Information File (S. R. HALL, N. SPADACCINI, I. D. BROWN, H. J. BERNSTEIN, J. D. WESTBROOK AND B. MCMAHON) 25
References 36
2.3. Specification of the Crystallographic Binary File (CBF/imgCIF) (H. J. BERNSTEIN AND A. P. HAMMERSLEY) 37
2.3.1. Introduction 37
2.3.2. CBF and imgCIF 37
2.3.3. Overview of the format 39
2.3.4. A complex example 42
2.3.5. imgCIF encodings 42
Appendix 2.3.1. Deprecated CBF conventions 43
References 43
2.4. Specification of the Molecular Information File (MIF) (F. H. ALLEN, J. M. BARNARD, A. P. F. COOK AND S. R. HALL) 44
2.4.1. Introduction 44
2.4.2. Historical background 44
2.4.3. MIF objectives 45
2.4.4. MIF concepts and syntax 45
2.4.5. Atoms, bonds and molecular representations 47
2.4.6. Bonding conventions 48
2.4.7. Structural templates 48
2.4.8. Stereochemistry and geometry at stereogenic centres 48

2.4.9. MIF query applications 50
2.4.10. Conclusion 51
References 51
2.5. Specification of the core CIF dictionary definition language (DDL1) (S. R. HALL AND A. P. F. COOK) 53
2.5.1. Introduction 53
2.5.2. The organization of a CIF dictionary 53
2.5.3. Definition attributes 54
2.5.4. DDL versions 54
2.5.5. The structure of DDL1 definitions 55
2.5.6. DDL1 attribute descriptions 57
References 60
2.6. Specification of a relational dictionary definition language (DDL2) (J. D. WESTBROOK, H. M. BERMAN AND S. R. HALL) 61
2.6.1. Introduction 61
2.6.2. The DDL2 presentation 61
2.6.3. Overview of the elements of DDL2 62
2.6.4. DDL2 organization 63
2.6.5. DDL2 dictionary applications 63
2.6.6. Detailed DDL2 specifications 65
References 70

2.1. Specification of the STAR File

BY S. R. HALL AND N. SPADACCINI

2.1.1. Introduction

A human language, in all its forms, is spoken, written and comprehended according to grammatical principles that evolve continuously with the need to express efficiently new ideas and experiences. In this chapter we will describe how similar principles have been developed to construct, describe and understand scientific data, such as numbers and codified text. The efficient and flexible expression of data may be achieved using grammatical rules similar to those of spoken languages, though the precise and unambiguous rendition and communication of data must preclude those nuances, subtleties and individual interpretation so important to the spoken and graphical world of poetry, literature and art. It is important to record these aspects of human endeavour, but they are distinctly different from the objectives of science, where information must be expressed with maximum precision and efficiency.

Understanding any language involves two fundamental steps – identification of the individual elements of a language sequence, and comprehension of the sequence structure. In a spoken language these steps normally comprise the simultaneous recognition of individual words and the understanding of their grammatical context. Such a simple decoding process belies the enormous potential for complexity in spoken languages. Nevertheless, most 'word sequences' are understood in this way and a similar approach may be used with non-textual data.

As discussed in Section 1.1.3, until quite recently most approaches to storing scientific data electronically were based on fixed-format structures. These are simple to construct and easy to comprehend provided the data layout is fixed and widely understood. That is, items, as well as lists of items, are written in a fixed sequential order that is mutually agreed on by those writing and reading the data. However, the preordained nature of a fixed format, which intentionally prevents changes to the data structure in a file, also poses a serious limitation for many scientific applications. This is because the nature of data used in scientific disciplines, such as in crystallography, evolves continuously and requires recording processes that are extensible and adaptable to change.

A lack of ready extensibility in the expression of data is particularly problematical for long-term archiving. For example, the recovery of information stored in a fixed format may be impossible if the layout details are lost or altered with time. Less rigid fixed-format variants, using keywords to identify groups of data, can improve the flexibility of data recording but they often preclude the introduction of new kinds of data.

In the 1980s it was widely recognized across the sciences that more general, extensible and expressive approaches were needed for recording, transmitting and archiving electronic data. This led to the development of free-format file structures. These file structures are the basis for universal file formats intended to: (*a*) store all kinds of data; (*b*) be independent of computer hardware (*i.e.* portable); (*c*) be both machine-parsable and human-understandable; (*d*) adapt to future data evolution (*i.e.* be extensible and robust); and (*e*) facilitate data structures of any complexity (*i.e.* have rich syntax capabilities).

The extent to which these objectives can be met determines the universality of a data-storage approach. Several of these properties are difficult to achieve simultaneously, and the compromises that have been adopted have largely determined the success of universal data languages in different fields. The STAR File is a universal data language applicable to all scientific disciplines, and the Crystallographic Information File described in Chapter 2.2 of this volume is a specific instance of the STAR format that has been adopted by the crystallographic community.

2.1.2. Universal data language concepts

As discussed above, the basic precepts of a universal data language parallel those of spoken languages in that a syntax and a grammar are used to describe and arrange (*i.e.* present) information, and hence define its comprehension. In particular, because data *values* (as numbers, symbols or text) are not often self-descriptive (*e.g.* a temperature value as a number alone cannot be distinguished from a number representing an atomic weight) interpretation depends critically on appropriate cues to the meaning and organization of data. A cue, in this context, has traditionally been prior knowledge of a fixed format, but can be explicit descriptions of data items, or even embedded codes identifying the relationship between data items. Cues represent the contextual thread of a data language, in the same way that grammatical rules provide words in natural-language sentences with meaning, and convert computer languages into executable code. This section will consider the concepts of data and languages that are important in understanding the syntax and purpose of a STAR File.

The first requirement of a language for electronic data transmission is that it be computer-interpretable, *i.e.* parsable. One should be able to read, interpret, manipulate and validate data entirely by machine. The other essential attributes of a universal file, as listed in Section 2.1.1, are flexibility, extensibility and applicability to all kinds of data. In modern-day science, data items change and expand continually and it is therefore paramount that a data language can accommodate new kinds of data seamlessly. It is the inflexibility and restrictive scope of fixed formats that limit their usefulness to crystallography for purposes other than local and temporary data retention.

2.1.2.1. Data models

Various approaches exist for meeting the objectives for a universal file listed in Section 2.1.1. The task is complicated by the fact that properties such as generality and simplicity, or accessibility and flexibility, are in many respects technically incongruent, and, depending on the nature of the data to be represented, particular compromises may be taken for efficiency or expediency. In other words, data languages, like spoken languages, are not equally efficient at expressing concepts, and the syntax of a language may need to be tailored to the target applications.

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; NICK SPADACCINI, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia.

This was one of the issues confronting the Working Party on Crystallographic Information formed by the IUCr in 1988 (Section 1.1.6) to decide on the most appropriate universal file data language for crystallography from those under development (McCarthy, 1990). It is interesting historically to note that one of these was HGML – not the web markup language we know today, but the *Human Genome Mapping Library* language. Another language considered by the working party was ASN.1 (ISO, 2002) used by the National Institute of Standards and Technology and several US Government departments. It is an accepted ANSI and ISO standard for data communication and is supported by software, such as NIST's OSI Toolkit. ASN.1 possesses a rich set of language constructs suited to representing complex data, but suffers from data identifiers that are encoded and not human-readable, and a syntax that is verbose (particularly for repetitive data items such as those common in crystallography). These characteristics mean that a typical Protein Data Bank (PDB) file expressed in ASN.1 notation increases in size by up to a factor of 5. This was hardly an attraction in the 1980s when storage media were very expensive. Moreover, the ASN.1 syntax is not particularly intuitive, and is difficult to read and to construct. In contrast, the STAR File proposed at the first WPCI meeting had a relatively simple syntax, was human-readable and provided a concise structure for repetitive data. It also proved suitable for constructing electronic dictionaries, as will be discussed in later chapters. However, its serious and well recognized weakness in 1988 was that any recording approach using a simple syntax to encode complex data must involve sophisticated parsing software, and at that time only the prototype software (*QUASAR*; Hall & Sievers, 1990) was available. It was therefore not a straightforward decision for the WPCI to decide to recommend the STAR File syntax as a more appropriate data language for crystallographic applications. It was this decision that led to the development of the CIF approaches described in this volume.

2.1.3. The syntax of the STAR File

The syntax of the STAR File (Hall, 1991; Hall & Spadaccini, 1994) has been used to develop a number of discipline-specific exchange and archival approaches, including the Crystallographic Information File (CIF) (Hall *et al*., 1991), the Molecular Information File (MIF) (Allen *et al*., 1995), the dictionary definition language (DDL1) (Hall & Cook, 1995), the macromolecular dictionary definition language (DDL2) (Westbrook & Hall, 1995) and the STAR dictionary definition language (StarDDL) (Spadaccini *et al*., 2000). The details of the CIF, MIF, DDL1 and DDL2 approaches are given in Chapters 2.2, 2.4, 2.5 and 2.6, respectively.

A STAR File is a sequential file containing lines of standard ASCII characters. A file may be divided into any number of discrete sets of unique data items. Sets may be in the form of data blocks, global blocks or save frames. The syntax rules for these sets are given below in descriptive form. A more rigorous description of the STAR File syntax is given in Appendix 2.1.1 in extended Backus–Naur form (McLennon, 1983).

The STAR File is a free-form language in which spaces (ASCII 32), vertical tabs (ASCII 11) and horizontal tabs (ASCII 9) are collectively referred to as <blank>, and newlines (ASCII 10), form feeds (ASCII 12) and carriage returns (ASCII 13) are collectively referred to as <terminate>. White spaces <wspace>, used to separate lexical tokens within the file, are all characters in the joined set of <blank> and <terminate>.

2.1.3.1. Text string

A text string is defined as any of the following.

(*a*) A sequence of non-white-space characters on a single line excluding a leading underscore <_> (ASCII 95).

Examples:

```
5.324
light-blue
```

(*b*) A sequence of characters on a single line containing the leading digraph <wspace><'> and the trailing digraph <'><wspace>. <'> is a single-quote character (ASCII 39) and <wspace> is white space.

Examples:

```
'light blue'
'classed as "unknown"'
'Patrick O'Connor'
```

Note that the use of the <'> character in the text string that is bounded by a <'> character is not precluded unless it is immediately followed by <wspace>. The leading and trailing digraphs serve to delimit the string and do not form part of the data. In the above example the value associated with the text field 'light blue' is **light blue**.

(*c*) A sequence of characters on a single line containing the leading digraph <wspace><"> and the trailing digraph <"><wspace>. <"> is a double-quote (ASCII 34) character and <wspace> is white space.

Examples:

```
"low melting point"
"Patrick O'Connor"
"Doug Collins' crystal"
"classed as "unknown""
```

The use of the <"> character in the text string that is bounded by a <"> character is not precluded unless it is immediately followed by <wspace>. The leading and trailing digraphs serve to delimit the string and do not form part of the data.

The text strings of type (*a*), (*b*) and (*c*) cannot contain line-breaking characters, and therefore cannot span multiple lines. There can be more than one text string per line if each value is preceded by a data name, or the values are part of a looped list (see Section 2.1.3.5).

(*d*) A sequence of lines starting with <terminate><;> and finishing with <terminate><;>, where <;> is the semicolon character (ASCII 59).

Example:

```
; School of CSSE
  UWA
;
```

The requirement that the <;> character be the first character on the line does not preclude the presence of other characters on the same line, in as much as it does not violate the STAR File structure.

The leading and trailing digraphs delimit the text field and do not form part of the data. The character sequence between the digraphs, including any line-breaking characters, constitutes the value of the text field. The value associated with the above example is <blank>**School**<blank>**of**<blank>**CSSE** <terminate><blank><blank>**UWA** (note in particular that the <terminate> character preceding the final ; delimiter is *not* part of the value).

2.1.3.2. Data name

A data name (or tag) is the identifier of a data value (see Section 2.1.3.3) and is a sequence of non-white-space characters starting with an underscore character `_` (ASCII 95).

Example:

```
_publication_author_address
```

2.1.3.3. Data value

A data value is a text string preceded by its identifying data name. Privileged keywords, such as described in Sections 2.1.3.5 to 2.1.3.8, are excluded from this definition.

2.1.3.4. Data item

A data item is a data value and its associated data name. Each data item stored in a STAR File is specified with this combination.

2.1.3.5. Data loop list

A looped list consists of the keyword `loop_` followed by

(*a*) a sequence of data names (possibly with nested `loop_` constructs); and

(*b*) a sequence of loop packets, each containing data values which are identified in the same order as the data names in (*a*).

A looped list specifies a table of data in which the data names represent the 'header descriptors' for columns of data and the packets represent the rows in the table. Looped lists may be nested to any level. Each loop level is initialized with the `loop_` keyword and is followed by the names of data items in this level. Data values that follow the nested data declarations must be in exact multiples of the number of data names. Each loop level must be terminated with a `stop_`, except the outermost (level 1) which is terminated by either a new data item or the privileged strings indicating a save frame (Section 2.1.3.6), a data block (Section 2.1.3.7), a global block (Section 2.1.3.8) or an end of file.

An example of a simple one-level loop structure is:

```
loop_
    _atom_identity_number
    _atom_type_symbol       1 C   2 C    3 O
```

Nested (multi-level) looped lists contain matching data packets [as per (*b*) above] and an additional `stop_` to terminate each level of data. Here is a simple example of a two-level nested list.

```
loop_
    _atom_id_number
    _atom_type_symbol
    loop_
        _atom_bond_id_1
        _atom_bond_id_2
        _atom_bond_order
             1 C     1 2 single  1 3 double stop_
             2 C     2 1 single             stop_
             3 O     3 1 double             stop_
```

The matching of data names to value packets is applied at each loop level. Initially the data values are matched to the data names listed in the outermost level loop. This process is iterated to successively inner levels. At the innermost loop level, data matching is maintained until a `stop_` is encountered. This returns the matching process to the next outer level. The matching process is recursive until the loop structure is depleted. Here is an example of a three-level loop structure.

```
loop_
    _atomic_name
     loop_
         _level_scheme
         _level_energy
          loop_
              _function_exponent
              _function_coefficient
 hydrogen
        (2)->[2]    -0.485813
        1.3324838E+01    1.0
        2.0152720E-01    1.0 stop_
        (2)->[2]    -0.485813
        1.3326990E+01    1.0
        2.0154600E-01    1.0 stop_
        (2)->[1]    -0.485813
        1.3324800E-01    2.7440850E-01
        2.0152870E-01    8.2122540E-01 stop_
        (3)->[2]    -0.496979
        4.5018000E+00    1.5628500E-01
        6.8144400E-01    9.0469100E-01
        1.5139800E-01    1.0000000E+01 stop_ stop_
```

2.1.3.6. Save frame

A save frame is a set of unique data items wholly contained within a data block. The frame starts with a `save_`*framecode* statement, where the *framecode* is a unique identifying code within the data block. Each frame is closed with a `save_` statement.

Example:

```
data_example
save_phenyl
    _object_class    molecular_fragment
     loop_
         _atom_identity_node
         _atom_identity_symbol 1 C 2 C 3 C 4 C 5 C 6 C
save_
loop_ _molecular_fragments   $ethyl  $phenyl $methyl
```

A save frame has the following attributes:

(*a*) A save frame may contain data items and loop structures but not other save frames [see (*f*)].

(*b*) The scope of the data specified in a save frame is the save frame in which it is specified.

(*c*) Data values in a save frame are distinct from any identical items in the parent data block.

(*d*) A save frame may be referenced within the data block in which it is specified using a data item with a value of `$`*framecode*.

Example:

```
loop_ _amino_acid_seq
      _amino_acid_data   1 $tyr 2 $arg 3 $arg 4 $leu
```

where 'arg', 'tyr' and 'leu' are frame codes identifying three save frames of data.

(*e*) A frame code must be unique within a data block.

(*f*) A save frame may not contain another save frame, but it may contain references to other save frames in the same data block using frame codes.

2.1.3.7. Data block

A data block is a set of data containing any number of unique items and save frames. A data block begins with a `data_`*blockcode* statement, where *blockcode* is a unique identifying name within a file. A data block is closed by another `data_`*blockcode* statement, a `global_` statement or an end of file.

Example:

```
data_rhinovirus
all information relevant to rhinovirus included here

data_influenza
all information relevant to influenza virus
included here
```

A data block has the following attributes:

(*a*) A block code must be unique within the file containing the data block.

(*b*) Data blocks may not be referenced from within a file [in contrast to save frames – see Section 2.1.3.6(*d*)].

(*c*) The scope of data specified in a data block is the data block. The value of a data item is always associated with the data block in which it is specified.

(*d*) Data specifications in a data block are unique, except they may be repeated within a save frame. Data specifications in a save frame are independent of the parent data block specifications.

(*e*) If a data item is not specified in a given data block, the global value is assumed. If a global value is not specified, the value is unknown.

2.1.3.8. Global block

A global block is a set of data items which are implied to be present in all data blocks which follow in a file, unless specified explicitly within a data block. A global block starts with a `global_` keyword and is closed by a `data_`*`blockcode`* statement or an end of file.

Example:

```
global_
information that is default within subsequent
data blocks
data_influenza
```

A global block has the following attributes:

(*a*) The scope of global data is from the point of declaration to the end of file.

(*b*) A global block may contain data items, loop structures and save frames.

(*c*) Multiple global blocks are concatenated to form a single block in which the last item specification has precedence.

(*d*) A data item specified within a data block has precedence over a data item specified in a prior global block.

2.1.3.9. Data sets and scopes

A data set is the generic term for a unique set of data. A STAR File may contain three types of data sets: global blocks, data blocks and save frames. The attributes of data sets are as follows.

(*a*) A file may contain any number of data sets.

(*b*) The data names defined within a data set must be unique to that set. That is, all `data_`*`blockcode`* names must be unique within the file, all data names must be unique within a `global_` block, all data names and `save_`*`framecode`*s must be unique within a data block, and all data names must be unique within a save frame.

(*c*) The scope of data sets is hierarchical (Fig. 2.1.3.1). Global blocks encompass all following data blocks; data blocks scope all contained save frames.

(*d*) The scope of a save frame is all data items contained within the frame.

(*e*) The scope of a data block is the boundaries of the data block, *i.e.* the end of the file or the start of the next data block, including any contained save frames. The same data item may be defined within a save frame and within the parent data block. All specifications of this item will be recognized when accessing the data block.

(*f*) The scope of a global block is the file, from the point of invocation to the end of file or the start of the next global block. It encompasses all contained global data items, data blocks and save frames. Globally specified data are active provided identical items are not specified in subsequent data sets.

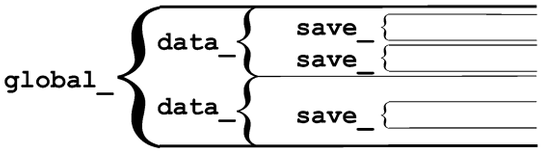


Fig. 2.1.3.1. Nested scopes of STAR data sets.

2.1.3.10. Privileged constructs

The following constructs are privileged.

(*a*) Text strings starting with the character sequences `data_`, `loop_`, `global_`, `save_` or `stop_` are privileged words (keywords) and may not be used as values in text strings of the type defined in Section 2.1.3.1(*a*).

(*b*) A sharp character `<#>` (ASCII 35) is an explicit end-of-line signal provided it is not contained within a text string of the types defined in Section 2.1.3.1(*b*), (*c*) or (*d*). Characters on the same line and following an active sharp character are considered as comment text.

2.1.3.11. Using stop_ in looped lists

In Section 2.1.3.5 we discuss how `stop_` is used to terminate a loop of data values and to return the looped list to the next outer nesting level. This same construction applies in the looped list of data names. The following, although not particularly intuitive, is a valid construction.

```
loop_
    _atom_id_number
        loop_
            _atom_bond_id_1
            _atom_bond_id_2
            _atom_bond_order  stop_
    _atom_type_symbol
                  1  1 2 single  1 3 double stop_ C
                  2  2 1 single             stop_ C
                  3  3 1 double             stop_ O
```

This is equivalent to the loop definition given in Section 2.1.3.5. One can use the `stop_` in name definitions to inhibit the nesting of loops in the definitions.

Appendix 2.1.1
Backus–Naur form of the STAR syntax and grammar

This description of the STAR syntax and grammar is annotated to clarify issues that cannot be represented in a pure extended Backus–Naur form (EBNF) definition.

The allowed character set in STAR is restricted to ASCII 09–13, 32–126. Other characters from the ASCII set are illegal. If such characters are present in a file the error state is well defined, but the functionality of the error handler is not specified. For instance, one may choose to return an illegal file exception and terminate the application or equally one may choose to ignore and skip over the illegal characters.

The concept of white space `<wspace>` includes a comment, since these only serve (in a parser sense) to delimit tokens anyway. We adopt the convention here of enclosing terminal symbols in single forward quotes. There are necessary provisos to this, and for representing formatting characters they are:

'`\'`' represents the single-quote character, *i.e.* the `\` is an escape character.

'\\' represents the single backslash character, *i.e.* the \ is an escape character.

'\f' represents the form-feed character, *i.e.* ASCII 12.

'\n' represents the new-line character, *i.e.* ASCII 10.

'\r' represents the carriage-return character, *i.e.* ASCII 13.

'\t' represents the tab character, *i.e.* ASCII 09.

'\v' represents the vertical-tab character, *i.e.* ASCII 11.

There are STAR specifications not definable in the EBNF. The EBNF can be used to define the tokenization of the input stream, and a STAR parser should test that the following condition is true. The number of data elements in <data_loop_values> of a <data_loop> production *must* be an integer multiple of the number of data names in the associated <data_loop_field>.

The STAR syntax specified in the EBNF follows.

A2.1.1.1. Lexical tokens

We accept a space, a horizontal and a vertical tab as <blank>.

```
<blank> ::= ' '| '\t' | '\v'  (ASCII 32 09 11)
```

The non-printing single ASCII characters 10, 12, 13 or any sequence of these are always interpreted as being a single line terminator. In this way there should not be operating-system-dependent ambiguity in those architectures that use character sequences as line terminators. This necessarily requires that these characters can only be used for line termination.

```
<terminate> ::= { '\n' | '\r' | '\f' }+
                (ASCII 10 13 12)
```

We define a 'comment' to be initiated with <blank> or <terminate> and the character #, followed by any sequence of characters (which include <blank>). The only characters not allowed are those in the production <terminate>, and hence these characters terminate a comment. Note the requirement of a leading <blank> or <terminate> is dropped if the # character is the first character in the file.

```
<comment> ::= { <blank> | <terminate> }+ '#' <char>*
```

We accept as white space *all* elements in the above three productions. White spaces are the lexemes able to delimit the lexical tokens. Note that a comment is a legitimate white space because it must end with a line terminator, and hence delimits tokens.

```
<wspace> ::= { <blank> | <terminate> | <comment> }+
```

Non-blank characters are composed of all the characters in our set, excluding <blank> and <terminate> characters.

```
<non_blank_char> ::= <ordinary_char> | '"' | '#' | '$'
                   | '\'' | ';' | '_' | '[' | ']'
```

<char> characters are composed of all the characters in our set, excluding <terminate> characters.

```
<char> ::= <blank> |  <non_blank_char>
```

We define a 'line of text' to be a line contained within a semicolon-bounded text block. Hence the first character *cannot* be a semicolon, and is followed by any number of characters from the set <char> and terminated with a line-termination character or just the termination character. This allows for 'blank' lines in the semicolon-bounded text block.

```
<line_of_text> ::=
     { <not_a_semi_colon> <char>* <terminate>
     | <terminate> }
```

Productions for specific characters.

```
<D_quote>       ::= '"'  (ASCII 34)
<S_quote>       ::= '\'' (ASCII 39)
<semi_colon>    ::= ';'  (ASCII 59)
```

All printable characters *except* the double quote.

```
<not_a_D_quote> ::= <ordinary_char> | '#' | '$' | '\''
                  | ';' | '_' | '[' | ']' | <blank>
```

All printable characters *except* the single quote.

```
<not_an_S_quote> ::= <ordinary_char> | '"' | '#' | '$'
                   | ';' | '_' | '[' | ']' | <blank>
```

All printable characters *except* the left and right square brackets.

```
<not_an_S_bracket> ::= <ordinary_char> | '"' | '#'
                     | '$' | ';' | '_'   | '\''
                     | <blank>
```

All printable characters *except* the semicolon.

```
<not_a_semi_colon> ::= <ordinary_char> | '"' | '#'
                     | '$' | '\'' | '_' | '[' | ']'
                     | <blank>
```

Ordinary characters are all those printable characters that can initiate a non-quoted text string. These exclude the special characters, ", #, $, ' and _ and in some cases ;.

```
<ordinary_char> ::=
       '!' | '%' | '&' | '(' | ')' | '*' | '+' | ','
     | '-' | '.' | '/' | '0' | '1' | '2' | '3' | '4'
     | '5' | '6' | '7' | '8' | '9' | ':' | '<' | '='
     | '>' | '?' | '@' | 'A' | 'B' | 'C' | 'D' | 'E'
     | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M'
     | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U'
     | 'V' | 'W' | 'X' | 'Y' | 'Z' | '\\' | '^' | '`'
     | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h'
     | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p'
     | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x'
     | 'y' | 'z' | '{' | '|' | '}' | '~' | <blank>
```

The keywords (in a case-insensitive form).

```
<DATA_> ::= {'d'|'D'} {'a'|'A'} {'t'|'T'} {'a'|'A'} '_'
<LOOP_> ::= {'l'|'L'} {'o'|'O'} {'o'|'O'} {'p'|'P'} '_'
<GLOBAL_> ::= {'g'|'G'} {'l'|'L'} {'o'|'O'} {'b'|'B'}
              {'a'|'A'} {'l'|'L'} '_'
<STOP_> ::= {'s'|'S'} {'t'|'T'} {'o'|'O'} {'p'|'P'} '_'
<SAVE_> ::= {'s'|'S'} {'a'|'A'} {'v'|'V'} {'e'|'E'} '_'
```

The operating-system-dependent end-of-file marker.

```
<EOF>   ::= end-of-file marker
```

A2.1.1.2. STAR grammar

A STAR File may be an empty file, or it may contain one or more data blocks or global blocks.

```
<STAR_File> ::=
     <wspace>* {<data_block> | <global_block>}*
```

There can be any amount of white spaces (remember <wspace> includes comments) before and at least one white space or an end of file (EOF) after a data or global block. This forces white space between data (and global) blocks in a single file. There must be *at least* one data item in any data or global block. This means a file consisting of just a data or global block heading is *invalid*.

```
<data_block>   ::= <data_heading>
                   {<data> | <save_block>}+
                   { <wspace>+ | <EOF> }
<global_block> ::= <GLOBAL_> {<data> | <save_block>}+
                   { <wspace>+ | <EOF> }
```

There can be any amount of white spaces (remember <wspace> includes comments) before a save-frame block. This forces white space between save-frame blocks also. There is no need to include the { <wspace>+ | <EOF> } found in data and global blocks, since those productions cover the situation of a save-frame block terminating the file.

```
<save_block> ::= <wspace>+ <save_heading>
                 <data>+ <wspace>+ <SAVE_>
```

A data-block or save-frame heading consists of the relevant five-character keyword (case-insensitive) immediately followed by at least one non-blank character. This does not preclude the associated block name or frame name consisting of just one or more punctuation characters.

```
<data_heading> ::= <DATA_> <non_blank_char>+
<save_heading> ::= <SAVE_> <non_blank_char>+
```

Data come in the following three forms.

(1) A data-name tag separated from its associated value by a trailing <blank>. Note it is explicitly a <blank> and not a <wspace>. These are *type I* data.

(2) A data-name tag separated from its associated value by a <terminate>. These are *type II* data.

(3) Looped data.

```
<data> ::= { <wspace>+ <data_name> <wspace>* <blank>
             <type_I_data_value> }
           | { <wspace>+ <data_name> <wspace>*
               <terminate> <type_II_data_value> }
           | <data_loop>
```

We must allow for white space preceding the loop_ (case-insensitive) keyword, since this is not covered by any of the other productions.

```
<data_loop> ::= <wspace>+ <LOOP_> <data_loop_field>
                <data_loop_values>
```

The name list for a loop must include at least one data name or a nested loop.

```
<data_loop_field> ::=
                  { <wspace>+ <data_name>
                  | <wspace>+ <LOOP_> <data_loop_field>
                    [<wspace>+ <STOP_>] }+
```

A data name is initiated by an underscore character and followed by one or more non-blank and non-terminating characters from the STAR character set. This does not preclude data names consisting of just one or more punctuation characters.

```
<data_name> ::= '_' <non_blank_char>+
```

Loop values are represented in the same way as the <data> production, except that the possibility of nested data loops introduces the need for the stop_ keyword.

```
data_loop_values> ::=
     { { <wspace>* <blank> <type_I_data_value> }
     | { <wspace>* <terminate> <type_II_data_value> }
     | <wspace>+ <STOP_> }+
```

Data values of type I data are immediately preceded by a <blank>. Data values of type II data are immediately preceded by a <terminate>.

```
<type_I_data_value> ::= <non_quoted_I_string>
     | { '"' <D_quote_string> '"' }
     | { '\'' <S_quote_string> '\'' }
     | { '[' <SB_bounded_string> ']' }

<type_II_data_value> ::= <non_quoted_II_string>
     | { '"' <D_quote_string> '"' }
     | { '\'' <S_quote_string> '\'' }
     | { ';' <SC_bounded_string> ';' }
     | { '[' <SB_bounded_string> ']' }
```

A type-I unquoted string is immediately preceded by a <blank>. It cannot begin with a number of characters (the complement of the <ordinary_char> set) *i.e.* ", #, $, ', [,] and _. However, it can begin with a semicolon. Then it is followed by any number of non-blank characters.

```
<non_quoted_I_string> ::=
     { <ordinary_char> | <semi_colon> }
     <non_blank_char>*
```

A type-II unquoted string is immediately preceded by a line break. As with type I, it too cannot begin with a ", #, $, ', [,] or _. It also *cannot* begin with a semicolon, since this would match the semicolon-delimited data production.

```
<non_quoted_II_string> ::=
     <ordinary_char> <non_blank_char>*
```

Specific exceptions to lexemes which match both types of unquoted strings are:

(1) No string beginning with an underscore is an unquoted string.

(2) No string that matches a production for <data_heading>, <save_heading>, <LOOP_>, <STOP_>, <SAVE_> or <GLOBAL_> is an unquoted string.

If one wishes to define data values which match lexemes excluded in cases (1) and (2) above, they should be *quoted* data values.

The string between a set of double quotes can consist of any character that is not a double quote, or it can be a double quote as long as it is immediately followed by a non-blank character or any number of double quotes at the end of the string. This final rule picks up cases of double-quote delimited strings that end in one or more double quotes, like "ABC"".

```
<D_quote_string> ::= { '"' <non_blank_char>
                     | <not_a_D_quote> }* { '"' }*
```

The string between a set of single quotes can consist of any character that is not a single quote, or it can be a single quote as long as it is immediately followed by a non-blank character or any number of single quotes at the end of the string. This final rule picks up cases of single-quote delimited strings that end in one or more single quotes, like 'ABC''.

```
<S_quote_string> ::= { '\'' <non_blank_char>
                     | <not_an_S_quote>}* { '\'' } *
```

The string bounded by semicolons can begin with any number of characters (including those in the <blank> production) but is necessarily terminated by a line break. This forces a line break on the line that contains the 'opening' semicolon. After the first line, one can have any number of <line_of_text>. Note we treat the first line as special, since it can contain a leading semicolon, which is not true of <line_of_text>. A <line_of_text> is always terminated with a line break, thus ensuring the closing semicolon is in column 1.

```
<SC_bounded_string> ::=
     <char>* <terminate> <line_of_text>*
```

The string bounded by square brackets can consist of any character including <terminate> and <blank>, and excluding the characters [and] unless they are escaped or are balanced.

```
<SB_bounded_string> ::= { <not_an_S_bracket>
                        | '\\' '[' | '\\' ']'
                        | <terminate>
                        | <SB_bounded_string> }*
```

The development of the STAR File, and particularly its application to crystallographic data, involved many people. The major contributions are acknowledged in the references below and other chapters in this volume. We name here only contributors who are not listed elsewhere in STAR-related publications. Particular thanks are due to Richard Goddard, Ted Maslen, Andre Authier, Philip Coppens, Jim King, Mike Dacombe, Peter Strickland, Mike Hoyland and George Sheldrick for their interest, support and commitment during the development of the STAR File concepts and its applications to crystallography.

References

Allen, F. H., Barnard, J. M., Cook, A. P. F. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Sievers, R. (1990). *QUASAR: CIF syntax checking and file manipulation tool.* ftp://ftp.iucr.org/pub/quasar.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

ISO (2002). ISO/IEC 8824-1. *Abstract Syntax Notation One (ASN.1). Specification of basic notation.* Geneva: International Organization for Standardization.

McCarthy, J. L. (1990). *Data interchange standards for biotechnology: issues and alternatives.* National Center for Biotechnical Information Report. NIH/DOE.

McLennon, B. J. (1983). *Principles of programming languages, design, evaluation and implementation*. New York: Holt, Rinehart and Winston.

Spadaccini, N., Hall, S. R. & Castleden, I. R. (2000). *Relational expressions in STAR File dictionaries. J. Chem. Inf. Comput. Sci.* **40**, 1289–1301.

Westbrook, J. D. & Hall, S. R. (1995). *A Dictionary Description Language for Macromolecular Structure.* http://ndbserver.rutgers.edu/mmcif/ddl.

2.2. Specification of the Crystallographic Information File (CIF)

BY S. R. HALL AND J. D. WESTBROOK
WITH SECTION 2.2.7 BY S. R. HALL, N. SPADACCINI, I. D. BROWN, H. J. BERNSTEIN, J. D. WESTBROOK AND B. MCMAHON

2.2.1. Introduction

The term 'Crystallographic Information File' (CIF) refers to data and dictionary files conforming to the conventions adopted by the IUCr in 1990 and revised by the IUCr Committee for the Maintenance of the CIF Standard (COMCIFS). The CIF format is intended to meet the needs of a wide range of scientific applications within, and without, the discipline of crystallography. Parts 2 and 3 of this volume provide the full specification of the contents of CIF across the different crystallographic applications. The files used in these applications must conform to the same rules of syntax, and share certain properties and conventions in the way that information is presented. It is these common features that are discussed in this chapter.

The CIF family of applications uses a proper subset of the STAR File syntax described in Chapter 2.1. The STAR File grammar provides a very general approach to storing and accessing data values through the use of an associated data name, or tag. A CIF search tool, such as *Star_Base* (Chapter 5.2), can readily access a single data value, or set of values, using this tag without prior knowledge of the order of the file contents. It can also provide details of the context of the data within the file structure. Context, in this sense, is a fully annotated indication of the file structure in which the retrieved value was located. That is, whether it was located in a global declaration, a named save frame, or a looped list. In every case the data 'value' is simply a character string and the STAR File protocol itself imposes absolutely no meaning on that string. This leaves the interpretation of the value string (*e.g.* whether it is numerical or text) to the conventions of the applications used to read and write the STAR File.

The CIF approach to the permissive STAR File syntax is restrictive. In the first place, the earliest version of the CIF syntax (Hall *et al.*, 1991) did not adopt some of the grammatical (or syntactical) constructs available to the STAR File in order to facilitate existing crystallographic software approaches. This was in anticipation of likely short-term developments, and to encourage a rapid take-up of the CIF approach. For these reasons it was considered appropriate to adopt only data-block partitioning and a single, rather than multiple, level of looped lists. Save frames were adopted into the CIF later but only in CIF dictionary files written using the DDL2 dictionary definition language (see Chapter 2.6). It is relevant to point out, however, that the full STAR File syntax has been adopted for the storage of NMR experimental and structure data (Ulrich *et al.*, 1998).

In 2002, a COMCIFS review of the design and implementation of CIF led to a revised syntax specification, which was published in February 2003. This revised specification is reproduced in full in Section 2.2.7. It is important to note that after more than a decade of CIF usage, this revision contains few substantial changes to the design choices of the original version (Hall *et al.*, 1991). There have been some modest extensions to the lengths of data names and text lines, and a number of clarifications are introduced. In addition, various privileged labels used for STAR File constructs (*e.g.* global blocks, save frames and nested loops, as described in Chapter 2.1) have now been explicitly reserved (*i.e.* excluded from appearing in unquoted form in an existing CIF). This will allow the clean upward migration of future CIF syntax versions to the more complex data structures permitted in a STAR File, when and if these are later required by the community.

The remainder of this chapter is structured as follows. First, there is a brief description of CIF terminology (Section 2.2.2). This is followed by the syntax rules, corresponding to a subset of the STAR syntax, used by CIF data files (Section 2.2.3). The portability and archival issues that programmers must be aware of in applying CIF data in different computing environments are described in Section 2.2.4. They are also detailed in the formal specifications given at the end of the chapter. Section 2.2.5 describes the conventions regarding data typing and embedded semantics that are common to all CIF applications, and Section 2.2.6 outlines future possible ways of introducing metadata which would enable files to be linked to each other, and which would establish the nature of CIF contents within a more general framework of information storage systems. The final section of the chapter, Section 2.2.7, reproduces in full the formal specification documents approved by COMCIFS.

2.2.2. Terminology

A summary of the basic terminology used throughout this chapter and the volume follows. A more extensive description of this terminology is given as formal specifications in Section 2.2.7.1.2.

(i) A **CIF** is a file conforming to the specification presented in this chapter. The term includes both **data files** containing information on a structural experiment or its results (or similar scientific content) and the **dictionary files** that provide descriptions of the data identifiers used in such data files.

(ii) A **data name** or **tag** is an identifier (a string of characters beginning with an underscore character) of the content of an associated data value.

(iii) A **data value** is a string of characters representing a particular item of information. It may represent a single numerical value; a letter, word or phrase; extended discursive text; or in principle any coherent unit of data such as an image, audio clip or virtual-reality object.

(iv) A **data item** is a specific piece of information defined by a data name and its associated data value.

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; JOHN D. WESTBROOK, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA; NICK SPADACCINI, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1; HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

(v) A **data block** is the highest-level component of a CIF, containing data items or (in the case of dictionary files only) save frames. A data block is identified by a **data-block header**, which is an isolated character string (that is, bounded by white space and not forming part of a data value) beginning with the case-insensitive reserved characters `data_`. A **block code** is the variable part of a data-block header, *e.g.* the string *foo* in the header `data_foo`.

(vi) A **looped list** of data is a set of data items represented as a table or matrix of values. The data names are assembled immediately following the word `loop_`, each separated by white space, and the associated data values are then listed in strict rotation. The table of values is assembled in row-major order; that is, the first occurrence of each of the data items is assembled in sequence, then the second occurrence of each item, and so forth. In a CIF, looped lists may not be nested.

2.2.3. The syntax of a CIF

The essential syntax rules for a CIF data file are discussed alongside an example (Fig. 2.2.3.1), which is an extract from a file used to exemplify the reporting of a small-molecule crystal structure to *Acta Crystallographica Section C*. The following discussion is tutorial in nature and is intended to give an overview of the syntactic features of CIF to the general reader. The special use of save frames in dictionary files is not discussed in this summary. Software developers will find the full specification at the end of the chapter. If there are any real or apparent discrepancies between the two treatments, the full specification is to be taken as definitive.

A CIF contains only ASCII characters, organized as lines of text.

Tokens (the discrete components of the file) are separated by white space; layout is not significant. Thus, in the list of atom-site coordinates in Fig. 2.2.3.1, the hydrogen-atom entries are cosmetically aligned in columns, but the non-aligned entries for the other atoms are equally valid. Indeed, there is no requirement that each cluster of looped data values be confined to a separate row; contrast the cosmetic ordering of the atom-sites loop with the loop of symmetry-equivalent positions, where entries run on the same or following lines indiscriminately.

A comment is a token introduced by a hash character # and extending to the end of the line. Comments are considered to have no portable information content and may freely be discarded by a parser. However, revision 1.1 of the CIF specification introduces a *recommendation* that a CIF begin with a comment taking the form

```
#\#CIF_1.1
```

where the 1.1 is a version identifier of the reference CIF specification. This is primarily for the benefit of general file-handling software on current operating systems (*e.g.* graphical file managers that associate software applications with files of specific type), and its presence or absence does not guarantee the integrity of the file with respect to any particular revision of the CIF specification.

The first non-comment token of a CIF must be a data-block header, which is a character string that does not include white space and begins with the case-insensitive characters `data_`.

The file may be partitioned into multiple data blocks by the insertion of further data-block headers. Data-block headers are case-insensitive (that is, two headers differing only in whether corresponding letter characters are upper or lower case are considered identical). Within a single data file identical data-block headers are not permitted.

```
data_99107abs

# Chemical data
_chemical_name_systematic
; 3-Benzo[b]thien-2-yl-5,6-dihydro-1,4,2-oxathiazine
  4-oxide
;
_chemical_formula_moiety         "C11 H9 N O2 S2"
_chemical_formula_weight          251.31

# Crystal data
_symmetry_cell_setting            orthorhombic
_symmetry_space_group_name_H-M    'P 21 21 21'

loop_
   _symmetry_equiv_pos_as_xyz
 'x, y, z' 'x+1/2, -y+1/2, -z' '-x, y+1/2, -z+1/2'
 '-x+1/2, -y, z+1/2'

_cell_length_a                    7.4730(11)
_cell_length_b                    8.2860(11)
_cell_length_c                    17.527(2)
_cell_angle_alpha                 90.00
_cell_angle_beta                  90.00
_cell_angle_gamma                 90.00

# Atomic coordinates and displacement parameters
loop_
  _atom_site_label
  _atom_site_type_symbol
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
S4 S 0.32163(7) 0.45232(6) 0.52011(3) 0.04532(13)
S11 S 0.39642(7) 0.67998(6) 0.29598(2) 0.04215(12)
O1 O -0.00302(17) 0.67538(16) 0.47124(8) 0.0470(3)
O4 O 0.2601(2) 0.28588(16) 0.50279(10) 0.0700(5)
N2 N 0.14371(19) 0.66863(19) 0.42309(9) 0.0402(3)
C3 C 0.2776(2) 0.57587(19) 0.43683(9) 0.0332(3)
C5 C 0.1497(3) 0.5457(3) 0.57608(11) 0.0498(5)
C6 C -0.0171(3) 0.5529(2) 0.52899(12) 0.0460(4)
C12 C 0.4215(2) 0.57488(19) 0.38139(9) 0.0344(3)
C13 C 0.5830(2) 0.4995(2) 0.38737(10) 0.0386(4)
C13A C 0.6925(2) 0.5229(2) 0.32123(10) 0.0399(4)
C14 C 0.8631(3) 0.4608(3) 0.30561(13) 0.0532(5)
C15 C 0.9423(3) 0.4948(3) 0.23709(15) 0.0644(7)
C16 C 0.8563(3) 0.5917(3) 0.18349(14) 0.0667(7)
C17 C 0.6901(3) 0.6568(3) 0.19729(12) 0.0546(5)
C17A C 0.6090(3) 0.6204(2) 0.26670(10) 0.0396(4)
H5A   H    0.1284   0.4834   0.6221   0.060
H5B   H    0.1861   0.6537   0.5908   0.060
H6A   H   -0.0374   0.4490   0.5050   0.055
H6B   H   -0.1186   0.5762   0.5617   0.055
H13   H    0.6182   0.4397   0.4297   0.046
H14   H    0.9218   0.3972   0.3414   0.064
H15   H    1.0548   0.4527   0.2262   0.077
H16   H    0.9127   0.6130   0.1373   0.080
H17   H    0.6340   0.7227   0.1616   0.066
```

Fig. 2.2.3.1. Typical small-molecule CIF.

Data names are character strings that begin with an underscore character _ and do not contain white-space characters. Data names serve to index data values and are case-insensitive.

Where a data name indexes a single data value, that value follows the data name separated by white space.

Where a data name indexes a set of data values (conceptually a vector or table column), the relevant data items are preceded by the case-insensitive string `loop_` separated by white space.

The examples of Fig. 2.2.3.1 show the use of `loop_` to specify a vector or one-dimensional list of values (the symmetry-equivalent positions) and a tabular or matrix list (the atom-site positions).

Again it should be emphasized that the rows and columns of the table are identified by parsing each value and referring back to the sequence in the header of its identifying tag, and not by a conventional layout of items. It is usual for CIF writers to lay out two-dimensional data arrays as well formatted tables for ease of inspection in text editors or other viewers, but CIF readers must never rely on layout alone to identify a tabulated value.

A corollary of this is that the number of values in a looped list must be an exact integer multiple of the number of data names declared in the loop header.

Within a single data block the same data name may not be repeated. Thus if a data item may have multiple values, these items must be collected together within a looped list, the data name itself being given once only in the loop header.

A data value is a string of characters. CIF distinguishes between numerical and character data in a broad sense (see Section 2.2.5.2 below). Numerical values may not contain white space (and indeed are constrained to a limited character set and ordering, essentially encompassing a small range of ways in which numbers are characteristically represented in printed form). Because tokens are separated by white space, character data that include white-space characters must be quoted. If the data value does not extend beyond the end of a line of text, it may be quoted by matching single-quote (apostrophe, `'`) or double-quote (quotation mark, `"`) characters. If the data value does extend beyond the end of a line of text, then paired semicolon characters `;` *as the first character of a line* may be used as delimiters. The closing semicolon is the first character of a line immediately following the close of the data-value string. Fig. 2.2.3.1 shows examples of all three types of delimited character values that include white space.

Character strings that begin with certain other characters must also be quoted. These leading characters are those which introduce tokens with special roles in a STAR File (such as underscore _ at the start of a data name, hash # at the start of a comment and dollar $ identifying a save-frame reference pointer). Likewise, the STAR File reserved words `loop_`, `stop_` and `global_` must be quoted if they represent data values, as must any character string beginning with `data_` or `save_`.

Lines of text are restricted to 2048 characters in length and data names are restricted to 75 characters in length. These are increases over the original values of 80 and 32 characters, respectively.

2.2.4. Portability and archival issues

The CIF format is designed to be independent of operating system (OS) and programming language. Nevertheless, variations in the way that each OS specifies and handles character sets mean that care must be taken to ensure that CIF software is portable across different computer platforms. There are also constraints on the application of these specifications in order to maintain compatibility between archival systems. These issues are discussed briefly here. More details are given in the formal CIF specification (see Section 2.2.7). In general, compatibility and portability considerations for different OSs are of little importance to users of CIFs, but they need to be well understood by software developers.

2.2.4.1. Character set

The characters permitted in a CIF are in effect the printable characters in the ASCII character set. However, a CIF may also be constructed and manipulated using alternative single-character byte mappings such as EBCDIC, and multi-byte or wide character encodings such as Unicode, provided there is a direct mapping to the permitted ASCII characters. Accented characters, characters in non-Latin alphabets and mathematical or special typographic symbols may not appear as single characters in a CIF, even if a host OS permits such representations.

White space (used to separate CIF tokens and within comments or quoted character-string values) is most portably represented by the printable space character (decimal value 32 in the ASCII character set). In an ASCII environment, white space may also be indicated by the control characters denoted HT (horizontal tab, ASCII decimal 9), LF (line feed, ASCII decimal 10) and CR (carriage return, ASCII decimal 13). To ease problems of translation between character encodings, the characters VT (vertical tab, ASCII decimal 11) and FF (form feed, ASCII decimal 12) are explicitly excluded from the CIF character set; this is a restriction that is not in the general STAR File specification (Chapter 2.1).

2.2.4.2. Line terminators

Given that the STAR File is built on the premise of a line-oriented text file, it is difficult in practice to provide a complete and portable description of how to identify the start or end of a line of text. The difficulty arises for two reasons.

First, some OSs or programming languages are record-oriented; that is, the OS is able to keep track of a region of memory associated with a specific record. It is usually appropriate to associate each such record with a line of text, but the 'boundaries' between records are managed by low-level OS utilities and are not amenable to a character-oriented discussion. Such systems, where records of fixed length are maintained, may also give rise to ambiguities in the interpretation of padding to the record boundary following a last printable character – is such padding to be discarded or treated as white space? It is for this reason that the elision of trailing white space in a line is permitted (but not encouraged) in the full CIF syntax specification [Section 2.2.7.1.4(17)].

The second complication arises because current popular OSs support several different character-based line terminators. Historically, applications developed under a specific OS have made general use of system libraries to handle text files, so that the conventions built into the system libraries have in effect become standard representations of line terminators for all applications built on that OS. For a long time this created no great problem, since files were transferred between OSs through applications software that could be tuned to perform the necessary line-terminator translations in transit. The best-known such application is undoubtedly the 'text' or 'ascii' transmission mode of the typical ftp (file transfer protocol) client.

Increasingly, however, a common network-mounted file system may be shared between applications running under different OSs and the same file may present itself as 'valid' to one user but not to another because of differences in what the applications consider a line terminator.

The problem of handling OS-dependent line termination is by no means unique to STAR File or CIF applications; any application that manipulates line-oriented text files must accommodate this difficulty. The specification notes the practice of designing applications that treat equivalently as a line terminator the characters LF (line feed or newline), CR (carriage return) or the combination CR followed by LF, since these are the dominant conventions under the prevailing Unix, MacOS and DOS/Windows OSs of the present day. While this will be a sensible design decision for many CIF-reading applications, software authors must be aware that the CIF specification aims for portability and archivability through a more general understanding of what constitutes a line of text.

2.2.4.3. Line lengths

The STAR File does not restrict the lengths of text lines. The original CIF specification introduced an 80-character limit to facilitate programming in Fortran and transmission of CIFs by email. Contemporary practices have enabled the line-length limit in the version 1.1 specification to be extended to 2048 characters. While the new limit in effect mandates the use of a 2048-character input buffer in compliant CIF-reading software, there is no obligation on CIF writers to generate output lines of this length.

2.2.4.4. Lengths of data names and block codes

CIF imposes another length restriction that is not integral to the STAR File syntax: data names, data-block codes and frame codes may not exceed 75 characters in length. This is an increase over the 32-character limit of the original specification, although this extension had already been approved by COMCIFS to coincide with the release of version 2 of the core dictionary (see Chapter 3.2).

There is no fundamental technical reason underlying this restriction; it permits assignment of a fixed-length buffer for recording such data names within a software application, but perhaps more significantly, it encourages a measure of conciseness in the creation of data names based on hierarchical component terms.

2.2.4.5. Case sensitivity

Following the general STAR File approach, the special tokens `loop_`, `save_`, the reserved word `global_`, data-block codes and save-frame codes, and data names are all case-insensitive. The case of any characters within data values must, however, be respected.

2.2.5. Common semantic features

As mentioned in Section 2.2.1, the STAR File structure allows retrieval of indexed data values without prior knowledge of the location of a data item within the file. Consequently there is no significance to the order of data items within a data block. However, molecular-structure applications will typically need to do more than retrieve an arbitrary string value. There is a need to identify the nature of individual data items (achieved portably through the definitions of standard data names in dictionary files), but also to be able to process the extracted data according to whether it is numerical or textual in nature, and possibly also to parse and extract more granular information from the entire data field that has been retrieved.

Different CIF applications have a measure of freedom to define many of the details of the content of data fields and the ways in which they may be processed – in effect, to define their semantic content. However, there are a number of conventions that are common to all CIF applications and this should be recognized in software applicable to a range of dictionaries.

These are discussed in some detail in the formal specification document in Section 2.2.7.4; in this section some introductory comments and additional explanations are given.

2.2.5.1. Data-name semantics

It is a fundamental principle of the STAR File approach that a data name is simply an arbitrary string acting as an index to a required value or set of values. It is equally legitimate to store the value of a crystal cell volume either as

```
_bdouiGFG78=z  1085.3(3)
```

or as

```
_cell_volume   1085.3(3)
```

provided that the users of the file have some way of discovering that the cell volume is indeed indexed by the tag `_bdouiGFG78=z` or `_cell_volume`, as appropriate.

However, it is conventional in CIF applications to define (in public dictionary files) data names that imply by their construction the meaning of the data that they index. Chapter 3.1 discusses the principles that are recommended for constructing data names and defining them in public dictionaries, and for utilizing private data names that will not conflict with those in the public domain.

Careful construction of data names according to the principles of Chapter 3.1 results in a text file that is intelligible to a scientist browsing it in a text editor without access to the associated dictionary definition files. In many ways this is useful; it allows the CIF to be viewed and understood without specialized software tools, and it safeguards some understanding of the content if the associated dictionaries cannot be found. On the other hand, there is a danger that well intentioned users may gratuitously invent data names that are similar to those in public use. It is therefore important for determining the correct semantic content of the values tagged by individual data names to make maximum possible disciplined use of the registry of public dictionaries, the registry of private dataname prefixes, and the facilities for constructing and disseminating private dictionaries discussed in Chapter 3.1.

2.2.5.2. Data typing

In the STAR File grammar, all data values are represented as character strings. CIF applications may define data types, and in the macromolecular (mmCIF) dictionary (see Chapter 3.6) a range of types has been assigned corresponding to certain contemporary computer data-storage practices (*e.g.* single characters, case-insensitive single characters, integers, floating-point numbers and even dates). This dynamic type assignment is supported by the relational dictionary definition language (DDL2; see Chapter 2.6) used for the mmCIF dictionary and is not available for all CIF applications.

However, a more restricted set of four primary or base data types is common to all CIF applications.

The type **numb** encompasses all data values that are interpretable as numeric values. It includes without distinction integers and non-integer reals, and the values may be expressed if desired in scientific notation. At this revision of the specification it does not include imaginary numbers. All numeric representations are understood to be in the number base 10.

It is, however, a complex type in that the standard uncertainty in a measured physical value may be carried along as part of the value. This is denoted by a trailing integer in parentheses, representing the integer multiple of the uncertainty in the last place of decimals in the numeric representation. That is, a value of '1085.3(3)' corresponds to a measurement of 1085.3 with a standard uncertainty of 0.3. Likewise, the value 34.5(12) indicates a standard uncertainty of 1.2 in the measured value.

Care should be taken in the placement of the parentheses when a number is expressed in scientific notation. The second example above may also be presented as 3.45E1(12); that is, the standard uncertainty is applied to the mantissa and not the exponent of the value.

Note that existing DDL2 applications itemize standard uncertainties as separate data items. Nevertheless, since the DDL2 dictionary includes the attribute `_item_type_conditions.code` with an allowed value of 'esd', future conformant DDL2 parsers might be expected to handle the parenthesized standard uncertainty representation.

The preferred behaviour of a CIF application is to determine the type of a data value by looking up the corresponding dictionary definition. However, some CIF-reading software may not be designed with the ability to parse dictionaries; and indeed any CIF reader may encounter data names that are not defined in a public or accompanying dictionary. It is therefore appropriate to adopt a strategy of interpreting as a number any data value that looks like one, *i.e.* adopts any of the permitted ways to represent a numeric value. Therefore, in the absence of a specific counter-indication (from a dictionary definition), the data value in the following example may be taken as the numeric (integer) value 1:

```
_unknown_data_name   1
```

On the other hand, if `_unknown_data_name` were explicitly defined in a dictionary with a data type of 'char', then the value should be stored as the literal character `1`.

This is a subtle point, perhaps of interest only to software authors. Nevertheless, the consistent behaviour of CIF applications will depend on correct implementation of this behaviour.

The data type **char** covers single characters or extended character strings. Since CIF tokens are separated by white space, any character string that includes white-space characters (including line-terminating characters) must be delimited by one or other of a set of special characters used for this purpose. The detailed rules for quoting such strings are given in Section 2.2.7.1.4 and comprise the standard CIF syntax rules for this case. No semantic distinction is made in general between short character strings and text strings that extend over several lines, described in the specification document as 'text fields', although again particular CIF applications may choose to impose distinctions. Note that numbers within a quoted string or a text block (bounded by semicolons in column 1) are not interpreted as type 'numb' but as type 'char'.

The data type **uchar** was introduced explicitly at revision 1.1 of the CIF specification, and is intended to formalize the description and automated handling of certain strings in CIFs that are case-insensitive (such as data names and data-block headers).

The data type **null** is a special type that has two uses. It is applied to items for which no definite value may be stored in computer memory. As such it is a formal device for allowing the introduction of data names into dictionary files that do not represent data values permissible within a data file instance. The usual example is that of the special data names introduced in DDL1 dictionaries (such as the core dictionary) to discuss categories.

The more important use of the null data type is its application to the meta characters '`?`' (query) and '`.`' (full point) that may occur as values associated with any data name and therefore have no specific type. (Arguably, for this case 'any' might be a better type descriptor than 'null'.)

The substitution of the query character '`?`' in place of a data value is an explicit signal that an expected value is missing from a CIF. This 'missing-value signal' may be used instead of omitting an item (*i.e.* its tag and value) entirely from the file, and serves as a reminder that the item would normally be present.

The substitution of the full-point character '`.`' in place of a CIF data value serves two similar, but not identical, purposes. If it is used in looped lists of data it is normally a signal that a value in a particular packet (*i.e.* a value in the row of the table) is 'inapplicable' or 'inappropriate'. In some CIF applications involving access to a data dictionary it is used to signal that the default value of the item is defined in its definition in the dictionary. Consequently, the interpretation of this signal is an application-specific matter and its use must be determined according to the application. For example, in a CIF submitted for publication in *Acta Crystallographica* the presence of a '`.`' value for the item `_geom_bond_site_symmetry_1` is predetermined as the default value 1_555 (as per the dictionary definition). Note that, in this instance, it is also equivalent to 'no additional symmetry' or 'inapplicable'.

2.2.5.3. Extended data typing: content type and encoding

The initial implementation of CIF assumed that most character strings would represent identifiers or terse descriptions or comments, and that the correct behaviour of the majority of CIF applications would be simply to store these in computer memory or retrieve them verbatim. Only a few data values were foreseen as having extended content that might need special handling. For example, the complete text of a manuscript was envisaged as being included in the field `_publ_manuscript_processed`. The handling of this field (its extraction and typesetting) would be left to unspecified external agents, although some clue as to the provenance of the contents of that field (and thus their appropriate handling) would be given by `_publ_manuscript_creation`.

However, the evolution of CIF applications has required that some element of typographic markup be permitted in a growing number of data values, and future applications may be envisaged in which graphical images, virtual-reality models, spreadsheet tables or other complex objects are embedded as the values of specific data items. Since it will not be possible to write general-purpose CIF applications capable of handling all such embedded content, techniques will need to be developed for transferring each such field to a specialized but separate content handler. In the meantime, the rather *ad hoc* conventions for introducing typographic markup available at present are described in Sections 2.2.7.4.13–17. It is hoped that in the future different types of such markup may be permitted so long as the data values affected can be tagged with an indication of their content type that allows the appropriate content handlers to be invoked.

It has also been necessary to allow native binary objects to be incorporated as CIF data values. This was done to support the storage of the large arrays of image data obtained from area detectors. Since the CIF character set is based on printable ASCII characters only, encodings including compression have been developed to permit interconversion between ASCII and binary representations of such data (see Chapter 2.3).

Nowadays, arbitrary embedded objects may be transported in web pages *via* the http protocol (Fielding *et al.*, 1999) or as attachments to email messages structured according to the MIME protocols (*e.g.* Freed & Borenstein, 1996). Identification of encoding techniques and hooks to invoke suitable handlers are carried in the relevant Content-Type and Content-Encoding http or MIME headers. It is suggested that this may form the basis of suitable tagging of content types and encoding for future CIF development.

A candidate for a CIF-specific encoding protocol is the special convention introduced with CIF version 1.1 to interconvert long lines of text between the new and old length limits (Section 2.2.7.4.11). This is an encoding in the sense that it is a device designed to retain any semantic content implicit in textual layout, while conforming to slightly different rules of syntax. It is designed to enable CIFs written to the longer line-length specification to be transformed so that they can still be handled by older software. Since the object of the exercise is to manage legacy applications, it is likely that the interconversion will be done through external applications, or filters, designed specifically for the purpose. Such a conversion filter is conceptually the same as a filter to convert a binary file into an ASCII base-64 encoding, for example.

2.2.6. CIF metadata and dictionary compliance

The development of several CIF dictionaries and fields of application has rapidly progressed beyond the specific purpose of describing a small-molecule or inorganic crystal structure for which CIF was devised. With these, a variety of application-specific metadata approaches have evolved to characterize the role of a particular CIF within a family of possible applications. These approaches use data definitions in dictionaries in which enumerated codes identify the file relationships. The mmCIF dictionary (see Chapter 3.6) allows informal identification of 'external reference files' which act as libraries of standard molecular geometry. The pdCIF dictionary (see Chapter 3.3) specifies identifiers that may be included within data blocks of external files containing calibration results. It is the responsibility of the file users to manage a lookup table or database between the referenced identifiers and the location of the files to which they pertain.

Two categories of data items currently exist in the core dictionary to allow a file to indicate its relationship to CIF dictionaries and other data files. (Equivalent categories are also present in the mmCIF dictionary.) AUDIT_CONFORM is a category of data names identifying the dictionaries that hold definitions of the data names in the current CIF. Particularly where the referenced dictionaries include any of the various public dictionaries described in Part 3 of this volume, this serves to establish the discipline within the broad fields of crystallography, structural biology and structural chemistry to which the data are most relevant.

The category AUDIT_LINK allows an informal textual description of the relationship between the data blocks within the current file. It is 'informal' in the sense that the relevant data items are free-text in nature. It would surely be useful to have a catalogue of more specific designations to allow automated software to track such relationships as the separate reference and modulated structures in an incommensurate compound, or the multiple trial refinements of a protein structure. The challenge is to determine and classify such standard relationships between data blocks.

In the future it is hoped that a common approach to metadata will be developed to enable all CIF instantiations to be uniquely identified and interrelated. Development of standard descriptions of the relationships between structural entities of this sort (reference geometries, calibration results, partial refinements, modulated superposed structures *etc.*) will be an important stage in the formalization of complete CIF metadata, and will become an important step towards categorization of data entities needed for interoperability between different file formats and across a wide range of scientific disciplines.

2.2.7. Formal specification of the Crystallographic Information File

Version 1.1 specification

BY S. R. HALL, N. SPADACCINI, I. D. BROWN,
H. J. BERNSTEIN, J. D. WESTBROOK AND B. MCMAHON

This section presents the documents *File syntax* (Sections 2.2.7.1–3) and *Common semantic features* (Section 2.2.7.4) that together comprise the formal CIF specification as approved by COMCIFS.

2.2.7.1. Syntax

2.2.7.1.1. *Introduction*

(1) This document describes the full syntax of the Crystallographic Information File (CIF).

2.2.7.1.2. *Definition of terms*

(2) The following terms are used in the CIF specification documents with the specific meanings indicated here.

(2.1) A **CIF** is a file conforming to the specification herein stated, containing either information on a crystallographic experiment or its results (or similar scientific content), or descriptions of the data identifiers in such a file.

(2.2) A **data file** is understood to convey information relating to a crystallographic experiment.

(2.3) A **dictionary file** is understood to contain information about the data items in one or more data files as identified by their data names.

(2.4) A **data name** is a case-insensitive identifier (a string of characters beginning with an underscore character) of the content of an associated data value.

(2.5) A **data value** is a string of characters representing a particular item of information. It may represent a single numerical value; a letter, word or phrase; extended discursive text; or in principle any coherent unit of data such as an image, audio clip or virtual-reality object.

(2.6) A **data item** is a specific piece of information defined by a data name and an associated data value.

(2.7) A **tag** is understood in this document to be a synonym for data name.

(2.8) A **data block** is the highest-level component of a CIF, containing data items or save frames. A data block is identified by a **data-block header**, which is an isolated character string (that is, bounded by white space and not forming part of a data value) beginning with the case-insensitive reserved characters `data_`.

(2.9) A **block code** is the variable part of a data-block header, *e.g.* the string `foo` in the header `data_foo`.

(2.10) A **save frame** is a partitioned collection of data items within a data block, started by a **save-frame header**, which is an isolated character string beginning with the case-insensitive reserved characters `save_`, and terminated with an isolated character string containing only the case-insensitive reserved characters `save_`.

(2.11) A **frame code** is the variable part of a save-frame header, *e.g.* the string `foo` in the header `save_foo`.

2.2.7.1.3. *File syntax*

(3) The syntax of CIF is a proper subset of the syntax of STAR Files as described by Hall (1991) and Hall & Spadaccini (1994). The general structure is described below in Section 2.2.7.1.4 and a number of subsections list specific restrictions to the STAR syntax that are in force within CIF. A formal language grammar using computer-science notation is included as Section 2.2.7.2.

2.2.7.1.4. *General features*

(4) A CIF consists of **data names** (tags) and associated values organized into **data blocks**. A data block may contain **data items** (associated data names and data values) and/or it may contain **save frames**.

(5) **Save frames** may only be used in dictionary files.

Implementation note: At a purely syntactic level there is no way to distinguish between dictionary and data files. (It is also to be noted that not all dictionary files contain save frames.) A fully validating parser must therefore be able to detect the start and termination of save frames, the uniqueness of the frame code within a data block and the uniqueness of data names within a frame code. It is, however, legitimate for an application-based parser designed to handle only the contents of data files to consider the presence of a save frame as an error.

(6) A **data block** begins with the reserved case-insensitive string `data_` followed immediately by the name of the data block, forming a **data-block header**. A **save frame** has a similar structure to a data block, but may not itself contain further save frames. A save frame begins with the reserved case-insensitive string `save_` followed immediately by the name of the save frame, forming a **save-frame header**. Unlike a data block, a save frame also has a marker for the end of the frame in the form of a repetition of the reserved case-insensitive word `save_`, this time without the name of the frame. Save frames may not nest. Within a single CIF, no two data blocks may have the same name; within a single data block no two save frames may have the same name, although a save frame may have the same name as a data block in the same CIF.

(7) A given **data name** (tag) [see (2.4) and (2.7)] may appear no more than once in a given data block or save frame. A tag may be followed by a single value, or a list of one or more tags may be marked by the preceding reserved case-insensitive word `loop_` as the headings of the columns of a table of values. White space is used to separate a data-block or save-frame header from the contents of the data block or save frame, and to separate tags, values and the reserved word `loop_`. Data items (tags along with their associated values) that are not presented in a table of values may be relocated along with their values within the same data block or save frame without changing the meaning of the data block or save frame. Complete tables of values (the table column headings along with all columns of data) may be relocated within the same data block or save frame without changing the meaning of the data block or save frame. Within a table of values, each tag may be relocated along with its associated column of values within the same table of values without changing the meaning of the table of values. In general, each row of a table of values may also be relocated within the same table of values without changing the meaning of the table of values. Combining tables of values or breaking up tables of values would change the meanings, and is likely to violate the rules for constructing such tables of values.

(8) The case-insensitive word `global_`, used in STAR Files to introduce a group of data values with a scope extending to the end of the file, is an additional reserved word in CIF (that is, it may not be used as the unquoted value of any data item).

(9) If a **data value** (2.5) contains white space or *begins* with a character string reserved for a special purpose, it *must* be delimited by one of several sets of special character strings (the choice of which is constrained if the data value contains characters interpretable as marking a new line of text according to the discussion in the following paragraphs). Such a data value will be indicated by the term *non-simple data value*.

(10) A *simple* data value (*i.e.* one which does not contain white space or begin with a special character string) may optionally be delimited by any of the same set of delimiting character strings, *except* for data values that are to be interpreted as numbers.

(11) The special character strings in this context are listed in the following table. The term 'non-simple data values' in this table refers to data values beginning with these special character strings.

Character or string	Role
`_`	identifies data name
`#`	identifies comment
`$`	identifies save-frame pointer
`'`	delimits non-simple data values
`"`	delimits non-simple data values
`[`	reserved opening delimiter for non-simple data values [see (19)]
`]`	reserved closing delimiter for non-simple data values [see (19)]
`;` (at the beginning of a line of text)	delimits non-simple data values
`data_`	identifies data-block header
`save_`	identifies save-frame header or terminator

In addition, the following case-insensitive *reserved words* may not occur as unquoted data values.

Reserved word	Role
`loop_`	identifies looped list of data
`stop_`	reserved STAR word terminating nested loops or loop headers
`global_`	reserved as a STAR global-block header

(12) The complete syntactic description of a numeric data value is included in Section 2.2.7.3(57) under the production (*i.e.* rule for constructing a part of the language) `<Numeric>`.

(13) *Comment:* The base CIF specification distinguishes between character and numeric values [see Section 2.2.7.4(15)]. Particular CIF applications may make more finely grained distinctions within these types. The paragraphs immediately above have the corollary that a data value such as `12` that appears within a CIF may be quoted (*e.g.* `'12'`) *if and only if* it is to be interpreted and stored in computer memory as a character string and not a numeric value. For example `'12'` might legitimately appear as a label for an atomic site, where another alphabetic or alphanumeric string such as `'C12'` is also acceptable; but it may *not* legitimately be used to represent an integer quantity twelve.

(14) Matching single- or double-quote characters (`'` or `"`) may be used to bound a string representing a non-simple data value *provided* the string does not extend over more than one line.

(15) *Comment:* Because data values are invariably separated from other tokens in the file by white space, such a quote-delimited character string may contain instances of the character used to delimit the string *provided* they are not followed by white space. For example, the data item

```
_example  'a dog's life'
```

is legal; the data value is `a dog's life`.

(16) *Comment:* Note that constructs such as

```
'an embedded \' quote'
```

do *not* behave as in the case of many current programming languages, *i.e.* the backslash character in this context does not escape the special meaning of the delimiter character. A backslash preceding the apostrophe or double-quote characters does, however, have special meaning in the context of accented characters (Section 2.2.7.4.15) provided there is no white space immediately following the apostrophe or double-quote character.

(17) The special sequence of end of line followed immediately by a semicolon in column one (denoted '`<eol>;`') may also be used as a delimiter at the beginning and end of a character string comprising a data value. The complete bounded string is called a **text field** and may be used to convey multi-line values. The end of line associated with the closing semicolon does *not* form part of the data value. Within a multi-line text field, leading white space within text lines must be retained as part of the data value; trailing white space on a line may however be elided.

(18) *Comment:* A text field delimited by the `<eol>;` digraph *may not* include a semicolon at the start of a line of text as part of its value.

(19) Matching square-bracket characters, '[' and ']', are *reserved* for possible future introduction as delimiters of multi-line data values. At this revision of the CIF specification, a data value may not begin with an unquoted left square-bracket character '['. (While not strictly necessary, the right square-bracket character ']' is restricted in the same way in recognition of its reserved use as a closing delimiter.)

(20) *Comment:* For example, the data value `foo` may be expressed equivalently as an unquoted string `foo`, as a quoted string `'foo'` or as a text field

```
;foo
;
```

By contrast, the value of the text field

```
; foo
  bar
;
```

is

```
 foo<eol>  bar
```

(where `<eol>` represents an end of line); the embedded space characters are significant.

(21) A comment in a CIF begins with an unquoted character '#' and extends to the end of the current line.

2.2.7.1.5. *Character set*

(22) Characters within a CIF are restricted to certain printable or white-space characters. Specifically, these are the ones located in the ASCII character set at decimal positions 09 (HT or horizontal tab), 10 (LF or line feed), 13 (CR or carriage return) and the letters, numerals and punctuation marks at positions 32–126.

Comment: The ASCII characters at decimal positions 11 (VT or vertical tab) and 12 (FF or form feed), often included in library implementations as white-space characters, are explicitly excluded from the CIF character set at this revision.

(23) *Comment:* The reference to the ASCII character set is specifically to identify characters in an established and widely available standard. It is understood that CIFs may be constructed and maintained on computer platforms that implement other character-set encodings. However, for maximum portability only the characters identified in the section above may be used. Other printable characters, even if available in an accessible character set such as Unicode, must be indicated by some encoding mechanism using only the permitted characters. At this revision, only the encoding convention detailed in Section 2.2.7.4(30)–(37) is recognized for this purpose.

2.2.7.1.6. *White space*

(24) Any of the white-space characters listed in paragraph (22) (*i.e.* HT, LF, CR) and the visible space character SP (position number 32 in the ASCII encoding) may be used interchangeably to separate tokens, with the exception that the semicolon characters delimiting multi-line text fields must be preceded by the white-space character or characters understood as indicating an end of line (see next paragraph).

2.2.7.1.7. *End-of-line conventions*

(25) The way in which a line is terminated is operating-system dependent. The STAR File specification does not address different operating-system conventions for encoding the end of a line of text in a text file. For a file generated and read in the same machine environment, this is rarely a problem, but increasingly applications on a network host may access files on different hosts through protocols designed to present a unified view of a file system. In practice, for current common operating systems many applications may regard the ASCII characters LF or CR or the sequence CR LF as signalling an end of line, inasmuch as these represent the end-of-line conventions supported under the common operating systems Unix, MacOS or DOS/Windows. On platforms with record-oriented operating systems, applications must understand and implement the appropriate end-of-line convention. Care must be taken when transferring such files to other operating systems to insert the appropriate end-of-line characters for the target operating system. A more complete discussion is given in (42) below.

2.2.7.1.8. *Case sensitivity*

(26) Data names, block and frame codes, and reserved words are case-insensitive. The case of any characters within data values must be respected.

2.2.7.1.9. *Implementation restrictions*

(27) Certain allowed features of STAR File syntax have been expressly excluded or restricted from the CIF implementation.

2.2.7.1.9.1. *Maximum line length and character set*

(28) Lines of text may not exceed 2048 characters in length. This count excludes the character or characters used by the operating system to mark the line termination.

The ASCII characters decimal 11 (VT) and 12 (FF) are excluded from the allowed character set [see paragraph (22)].

2.2.7.1.9.2. *Maximum data-name, block-code and frame-code lengths*

(29) Data names may not exceed 75 characters in length.

(30) Data-block codes and save-frame codes may not exceed 75 characters in length (and therefore data-block headers and save-frame headers may not exceed 80 characters in length).

2.2.7.1.9.3. *Single-level loop constructs*

(31) Only a single level of looping is permitted.

2.2.7.1.9.4. *Non-expansion of save-frame references*

(32) Save frames are permitted in CIFs, but expressly for the purpose of encapsulating data-name definitions within data dictionaries. No reference to these save frames is envisaged, and the save-frame reference code permitted in STAR is not used. This means that unquoted character strings commencing with the `$` character may not be interpreted as save-frame codes in CIF. Use of such unquoted character strings is *reserved* to guard against subsequent relaxation of this constraint.

2.2.7.1.9.5. *Exclusion of global_ blocks*

(33) In the full STAR specification, blocks of data headed by the special case-insensitive word `global_` are permitted before normal data blocks. They contain data names and associated values which are inherited in subsequent data blocks; the scope of a value extends from its point of declaration in a global block to the end of the file. Because rearrangements of the order of data blocks and concatenation of data blocks from different files are commonplace operations in many CIF applications, and because of the difficulty in properly tracking and implementing values implied by global blocks, use of the `global_` feature of STAR is expressly *forbidden* at this revision. To guard against its future introduction, the special case-insensitive word `global_` remains *reserved* in CIF.

Table 2.2.7.1. *A formal grammar for CIF*

(*a*) Basic structure of a CIF.

Syntactic unit	Syntax	Case sensitive?
<CIF>	<Comments>? <WhiteSpace>? {<DataBlock> {<WhiteSpace> <DataBlock> }* {<WhiteSpace> }? }?	yes
<DataBlock>	<DataBlockHeading> {<WhiteSpace> { <DataItems> | <SaveFrame>} }*	yes
<DataBlockHeading>	<DATA_> {<NonBlankChar> }+	no
<SaveFrame>	<SaveFrameHeading> {<WhiteSpace> <DataItems> }+ <WhiteSpace> <SAVE_>	yes
<SaveFrameHeading>	<SAVE_> {<NonBlankChar> }+	no
<DataItems>	<Tag> <WhiteSpace> <Value> | <LoopHeader> <LoopBody>	yes
<LoopHeader>	<LOOP_> {<WhiteSpace> <Tag>}+	no
<LoopBody>	<Value> {<WhiteSpace> <Value> }*	yes

(*b*) Reserved words.

Syntactic unit	Syntax	Case sensitive?
<DATA_>	{'D'|'d'} {'A'|'a'} {'T'|'t'} {'A'|'a'} '_'	no
<LOOP_>	{'L'|'l'} {'O'|'o'} {'O'|'o'} {'P'|'p'} '_'	no
<GLOBAL_>	{'G'|'g'} {'L'|'l'} {'O'|'o'} {'B'|'b'} {'A'|'a'} {'L'|'l'} '_'	no
<SAVE_>	{'S'|'s'} {'A'|'a'} {'V'|'v'} {'E'|'e'} '_'	no
<STOP_>	{'S'|'s'} {'T'|'t'} {'O'|'o'} {'P'|'p'} '_'	no

(*c*) Tags and values.

Syntactic unit	Syntax	Case sensitive?
<Tag>	'_'{<NonBlankChar>}+	no
<Value>	{'.' | '?' | <Numeric> | <CharString> | <TextField> }	yes

(*d*) Numeric values.

Syntactic unit	Syntax	Case sensitive?
<Numeric>	{<Number> | <Number> '(' <UnsignedInteger> ')' }	no
<Number>	{<Integer> | <Float> }	no
<Integer>	{{ '+' | '-' }? <UnsignedInteger>	no
<Float>	{ <Integer><Exponent> | { {'+'|'-'} ? { {<Digit>} * '.' <UnsignedInteger> } | { <Digit>} + '.' } } {<Exponent>} ? } }	no
<Exponent>	{ {'e' | 'E' } | {'e' | 'E' } { '+' | '-' } } <UnsignedInteger>	no
<UnsignedInteger>	{<Digit> }+	no
<Digit>	{ '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' }	no

(*e*) Character strings and text fields.

Syntactic unit	Syntax	Case sensitive?
<CharString>	<UnquotedString> | <SingleQuotedString> | <DoubleQuotedString>	yes
<eol><UnquotedString>	<eol><OrdinaryChar> {<NonBlankChar>}*	yes
<noteol><UnquotedString>	<noteol>{<OrdinaryChar>|';'} {<NonBlankChar>}*	yes
<SingleQuotedString><WhiteSpace>	<single_quote>{<AnyPrintChar>}* <single_quote> <WhiteSpace>	yes
<DoubleQuotedString><WhiteSpace>	<double_quote> {<AnyPrintChar>}* <double_quote> <WhiteSpace>	yes
<TextField>	{<SemiColonTextField> }	yes
<eol><SemiColonTextField>	<eol>';' { {<AnyPrintChar>}* <eol> {{<TextLeadChar> {<AnyPrintChar>}*}? <eol>}* } ';'	yes

Table 2.2.7.1. *(cont.)*

(*f*) White space and comments.

Syntactic unit	Syntax	Case sensitive?
<WhiteSpace>	{<SP>|<HT>|<eol>|<TokenizedComments>}+	yes
<Comments>	{ '#' {<AnyPrintChar>}* <eol>}+	yes
<TokenizedComments>	{<SP>|<HT>|<eol>|}+ <Comments>	yes

(*g*) Character sets.

Syntactic unit	Syntax	Case sensitive?
<OrdinaryChar>	{ '!'|'%'|'&'|'('|')'|'*'|'+'|','|'-'|'.'|'/'|'0'|'1'|'2'|'3'|'4'|'5'| '6'|'7'|'8'|'9'|':'|'<'|'='|'>'|'?'|'@'|'A'|'B'|'C'|'D'|'E'|'F'|'G'|'H'| 'I'|'J'|'K'|'L'|'M'|'N'|'O'|'P'|'Q'|'R'|'S'|'T'|'U'|'V'|'W'|'X'|'Y'|'Z'| '\'|'^'|'`'|'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|'i'|'j'|'k'|'l'|'m'|'n'|'o'| 'p'|'q'|'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'|'{'|'|'|'}'|'~' }	yes
<NonBlankChar>	<OrdinaryChar>|<double_quote>|'#'|'$'|<single_quote>|'_' |';'|'['|']'	yes
<TextLeadChar>	<OrdinaryChar>|<double_quote>|'#'|'$'|<single_quote>|'_'|<SP>|<HT>|'['|']'	yes
<AnyPrintChar>	<OrdinaryChar>|<double_quote>|'#'|'$'|<single_quote>|'_'|<SP>|<HT>|';'|'['|']'	yes

2.2.7.1.10. *Version identification*

(34) As an archival file format, the CIF specification is expected to change infrequently. Revised specifications will be issued to accompany each substantial modification. A CIF may be considered compliant against the most recent version for which in practice it satisfies all syntactic and content rules as detailed in the formal specification document. However, to signal the version against which compliance was claimed at the time of creation, or to signal the file type and version to applications (such as operating-system utilities), it is *recommended* that a CIF begin with a structured comment that identifies the version of CIF used. For CIFs compliant with the current specification, the first 11 bytes of the file should be the string

```
#\#CIF_1.1
```

immediately followed by one of the white-space characters permitted in paragraph (22).

2.2.7.2. A formal grammar for CIF

2.2.7.2.1. *Summary*

(35) The rows of Table 2.2.7.1 are called 'productions'. Productions are rules for constructing sentences in a language. They are written in terms of 'terminal symbols' and 'non-terminal symbols'. 'Terminal symbols' are what actually appear in a language. For example, `'poodle'` might be given as a string of terminal symbols in some language discussing dogs. Non-terminal symbols are the higher-level constructs of the language, *e.g.* sentences, clauses, *etc*. For example `<DOG>` might be given as a non-terminal symbol in some language discussing dogs. Productions may be used to infer rules for parsing the language. For example,

```
<DOG> ::= { 'poodle'|'terrier'|'bulldog'|'greyhound' }
```

might be given as a rule telling us what names of types of dogs we are allowed to write in this language. In this table, terminal symbols (*i.e.* terminal character strings) are enclosed in single quotes. To avoid confusion, the terminal symbol consisting of a single quote (*i.e.* an apostrophe) is indicated by `<single_quote>` and the terminal symbol consisting of a double quote is indicated by `<double_quote>`. The printable space character is indicated by `<SP>`, the horizontal tab character by `<HT>` and the end of a line by `<eol>`. To allow for the occurrence of a semicolon as the initial character of an unquoted character string, provided it is not the first character in a line of text, the special symbol `<noteol>` is used below to indicate any character that is not interpretable as a line terminator. The cases of context sensitivity involving the beginning of text fields and the ends of quoted strings are discussed below, but they are most commonly resolved in a lexical scan.

(36) Productions can be used to produce documents, or equivalently to check a document to see if it is valid in this grammar. The angle brackets delimit names for the syntactic units (the 'non-terminal symbols') being defined. The curly braces enclose alternatives separated by vertical bars and/or followed by a plus sign for 'one or more', an asterisk for 'zero or more' or a question mark for 'zero or one'.

(37) In most cases, each production has a single non-terminal symbol in the syntactic unit being defined. However, in some cases, both the syntactic unit and the syntax begin or end with some common symbol. This indicates that a specific context is required in order for the rule to be applied. This is done because the initial semicolon of a semicolon-delimited text field only has meaning at the beginning of a line, and quoted strings may contain their initial quoting character provided the embedded quoting character is not immediately followed by white space. This 'context-sensitive' notation is unusual in defining computer languages (although very common in the full specifications of many computer and non-computer languages). This context-sensitive notation greatly simplifies the definitions and is simple to implement. The formal definitions are elaborated below.

(38) In the present revision, the production for `<TextField>` is a trivial equivalence to `<SemiColonTextField>`. The redundancy is retained to permit possible future extensions to text fields, in particular the possible introduction of a bracket-delimited text value.

2.2.7.2.2. *Explanation of the formal syntax*

Comment: Readers not familiar with the conventions used in describing language grammars may wish to consult various lecture notes on the subject available on the web, *e.g.* Bernstein (2002).

(39) In creating a parser for CIF, the normal process is to first perform a 'lexical scan' to identify 'tokens' in the CIF. A 'token' is a grammatical unit, such as a special character, or a tag or a value, or some major grammatical subunit. In the course of a lexical scan, the input stream is reduced to manageable pieces, so that

the rest of the parsing may be done more efficiently. The convention followed in this document is to mark the 'non-terminal' tokens that are built up out of actual strings of characters or which do not have an immediate representation as printable characters by angle brackets, <>, and to indicate the tokens that are actual strings of characters as quoted strings of characters.

(40) The precise division between a lexical scan and a full parse is a matter of convenience. A suggested division is presented. Before getting to that point, however, there are some highly machine-dependent matters that need to be resolved. There must be a clear understanding of the character set to be used, and of how files and lines begin and end. The character set will be specified in terms of printable characters and a few control characters from the 7-bit ASCII character set. In addition, we will need some means of specifying the end of a line.

(41) The character set in CIF is restricted to the ASCII control characters <HT> (horizontal tab, position 09 in the ASCII character set), <NL> (newline, position 10 in the ASCII character set, also named <LF>) and <CR> (carriage return, position 13 in the ASCII character set), and the printable characters in positions 32–126 of the ASCII character set. These are the characters permitted by STAR with the exception of VT (vertical tab, position 11 in the ASCII character set) and FF (form feed, position 12 in the ASCII character set). In general it is poor practice to use characters that are not common to all national variants of the ISO character set. On systems or in programming languages that do not 'work in ASCII', the characters themselves may have different numeric values and in some cases there is no access to all the control characters.

(42) The <eol> token stands for the system-dependent end of line.

Implementation note: CIF implementations may follow common HTML and XML practice in handling <eol>:

> '[On many modern systems,] lines are typically separated by some combination of the characters carriage-return (#xD) and line-feed (#xA). To simplify the tasks of applications, the characters passed to an application ... must be as if the ... [parser] normalized all line breaks in external parsed entities ... on input, before parsing, [*e.g.*] by translating both the two-character sequence #xD #xA and any #xD that is not followed by #xA to a single #xA character.'

(From the XML specification http://www.w3.org/TR/2000/REC-xml-20001006.)

Because Unix systems use \n (the ASCII LF control character, or #xA), MS Windows systems use \r\n (the ASCII CR control character, or #xD, followed by the ASCII LF control character, or #xA) and classic MacOS systems use \r, a parser which covers a wide range of systems in a reasonable manner could be constructed using a pseudo-production for <eol> such as

```
<eol> ::= { <LF> | <CR><LF> | <CR> }
```

provided the supporting infrastructure (such as the lexer) deals with the necessary minor adjustment to ensure that each end of line is recognized and that all end-of-line control characters are filtered out from the portions of the text stream that are to be processed by other productions. One case to handle with care is the end-of-document case. It is not uncommon to encounter a last line in a document that is not terminated by any of the above-mentioned control characters. Instead, it may be terminated by the end of the character stream or by a special end-of-text-document control character [*e.g.* #x4 (control-D) or #x1A (control-Z)]. A CIF parser should normalize such unterminated terminal lines to appear to an application as if they had been properly terminated. On the other hand, care should also be taken so that in multiple generations of CIF processing such processing does not result in an ever-growing 'tail' of empty lines at the end of a CIF document.

This discussion is *not* meant to imply that a parser for a system that uses one of these line-termination conventions must recognize a CIF written using another of these line-termination conventions.

This discussion is *not* meant to imply that parsers on systems that use other line-termination conventions and/or non-ASCII character sets need to handle these ASCII control characters.

In processing a valid CIF document, it is always sufficient that a parser be able to recognize the line-termination conventions of text files local to its system environment, and that it be able to recognize the local translations of <SP><HT> and the printable characters used to construct a CIF.

However, when circumstances permit, if a parser is able to recognize 'alien' line terminations, it is permissible for the parser to accept and process the CIF in that form without treating it as an error.

In writing CIF documents, the software that emits lines should follow the text-file line-termination conventions of the target system for which it is writing the CIF documents, and not mix conventions from multiple systems. In transmitting a CIF document from system to system, software should be used that causes the document to conform to the line-termination conventions of the target system. In most cases this objective can best be achieved by using 'text' or 'ascii' transmission modes, rather than 'binary' or 'image' transmission modes.

(43) In order to write the grammar, we need a way to refer to the single-quote characters which we use both to quote within the syntax and to quote within a CIF. To avoid system-dependent confusion, we define the following special tokens:

Token	Meaning
<SP>	' ', the printable space character
<HT>	the horizontal-tab character on the system
<eol>	the machine-dependent end of line
<noteol>	the complement of the above; any character that does not indicate the machine-dependent end of line
<single_quote>	the apostrophe, '
<double_quote>	the double-quote character, "

(44) There are CIF specifications not definable directly in a context-free Backus–Naur form (BNF). Restrictions in record and data-name lengths, and the parsing of text fields and quoted character strings are best handled in the initial lexical scan. A pure BNF can then be used to parse the tokenized input stream.

2.2.7.3. Lexical tokens

(45) We define a 'comment' to be initiated with the character #. This can be followed by any sequence of characters (which include <SP> or <HT>). The only characters not allowed are those in the production <eol>, which <eol> terminates a comment. A comment is recognized only at the beginning of a line or after blanks, *i.e.* only after space, tab or <eol>. For this reason we define both comments and 'tokenized comments'. No portion of the essential machine-readable content within a CIF is conveyed by the comments. Comments are for the convenience of human readers of CIFs and may be freely introduced or removed. Note however the optional structured comment sanctioned in paragraph (34) above, which has the purpose of indicating the file type and revision level to general-purpose file-handling software.

```
<Comments>          ::= { '#' {<AnyPrintChar>}* <eol>}+
<TokenizedComments> ::= { <SP>|<HT>|<eol> }+ <Comments>
```

(46) We accept as white space all appropriate combinations of spaces, tabs, end of lines and comments, as well as the beginning of the file. White space are the characters able to delimit the lexical tokens.

```
<WhiteSpace> ::= {<SP>|<HT>|<eol>|<TokenizedComments>}+
```

(47) Non-blank characters are composed of all the characters in our set, excluding <SP> and <HT> and <eol> characters.

```
<NonBlankChar> ::= <ordinary_char>|<double_quote>|'#'|
     '$'|<single_quote>|'_'|';'|'['|']'
```

(48) AnyPrintChar characters are composed of all the characters in our set excluding <eol> characters.

```
<AnyPrintChar> ::= <ordinary_char>|<double_quote>|'#'|
     '$'|<single_quote>|'_'|<SP>|<HT>|';'|'['|']'
```

(49) We define a 'line of text' to be a line contained within a semicolon-bounded text field. Hence the first character *cannot* be a semicolon; it may be followed by any number of characters from the set <char> and terminated with a line-termination character. We define the characters in <TextLeadChar> as those in <AnyPrintChar> except for the semicolon.

```
<TextLeadChar> ::= <ordinary_char>|<double_quote>|'#'|
     '$'|<single_quote>|'_'|<SP>|<HT>|'['|']'
```

(50) Ordinary characters are all those printable characters that can initiate a non-quoted character string. These exclude the special characters ", #, $, ', [,] and _, and in some cases ;.

```
<OrdinaryChar> ::=
      '!'|'%'|'&'|'('|')'|'*'|'+'|','|'-'|'.'|'/'|'0'
     |'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'|':'|'<'|'='
     |'>'|'?'|'@'|'A'|'B'|'C'|'D'|'E'|'F'|'G'|'H'|'I'
     |'J'|'K'|'L'|'M'|'N'|'O'|'P'|'Q'|'R'|'S'|'T'|'U'
     |'V'|'W'|'X'|'Y'|'Z'|'\'|'^'|'`'|'a'|'b'|'c'|'d'
     |'e'|'f'|'g'|'h'|'i'|'j'|'k'|'l'|'m'|'n'|'o'|'p'
     |'q'|'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'|'{'|'|'
     |'}'|'~'
```

(51) The *reserved word* **data_** (in a case-insensitive form).

```
<DATA_> ::= {'d'|'D'} {'a'|'A'} {'t'|'T'} {'a'|'A'} '_'
```

(52) The *reserved word* **loop_** (in a case-insensitive form).

```
<LOOP_> ::= {'l'|'L'} {'o'|'O'} {'o'|'O'} {'p'|'P'} '_'
```

(53) The *reserved word* **save_** (in a case-insensitive form).

```
<SAVE_> ::= {'s'|'S'} {'a'|'A'} {'v'|'V'} {'e'|'E'} '_'
```

(54) The *reserved word* **stop_** (in a case-insensitive form).

```
<STOP_> ::= {'s'|'S'} {'t'|'T'} {'o'|'O'} {'p'|'P'} '_'
```

(55) The *reserved word* **global_** (in a case-insensitive form). This is actually a reserved word of STAR, but we define it here so that it may be explicitly excluded as an unquoted string. We do this so that any possible future adoption of STAR features will not invalidate existing CIFs.

```
<GLOBAL_> ::= {'g'|'G'} {'l'|'L'} {'o'|'O'} {'b'|'B'}
              {'a'|'A'} {'l'|'L'} '_'
```

(56) Quoted strings need to be recognized in the lexical scan, because their definition is context-sensitive. A string quoted by single quotes may contain a single quote as long as it is not followed by white space. A string quoted by double quotes may contain a double quote as long as it is not followed by white space. Formally we express this with context-sensitive productions. In practice, it requires a one-character look-ahead to decide to continue the scan if the opening quote is encountered, but the following character is not space, tab or end of line. When processing a semicolon-delimited text field, the column position has to be remembered to decide whether a semicolon should be recognized.

For a semicolon-delimited text string, failure to provide trailing white space is an error. The <WhiteSpace> on the left-hand side must evaluate to the same string instance on the right-hand side and the parse must terminate on the first valid match reading left to right.

```
<SingleQuotedString><WhiteSpace> ::=
     <single_quote>{<AnyPrintChar>}*<single_quote>
     <WhiteSpace>
<DoubleQuotedString><WhiteSpace> ::=
     <double_quote>{<AnyPrintChar>}*<double_quote>
     <WhiteSpace>
<TextField> ::= {<SemiColonTextField>}
<eol><SemiColonTextField> ::=
     <eol>';' { {<AnyPrintChar>}* <eol>
          {{<TextLeadChar> {<AnyPrintChar>}*}? <eol>}*
           } ';'
<BracketTextField> ::=
     '[' { <NonBracketChar>|<BracketTextField>}* ']'
```

(57) Tags and values are appropriate lexical tokens. The special values of '.' and '?' represent data that are inapplicable or unknown, respectively.

(i) No string that matches the production for <LOOP_> is accepted as a non-quoted string.

(ii) No string that matches the production for <STOP_> is accepted as a non-quoted string.

(iii) No string in which the initial five characters match the production for <DATA_> is accepted as a non-quoted string.

(iv) No string in which the initial five characters match the production for <SAVE_> is accepted as a non-quoted string.

(v) No string that matches the production for <GLOBAL_> is accepted as a non-quoted string.

Unquoted strings are described by a pair of productions to permit the initial letter of an unquoted string to be a semicolon so long as that does not occur at the beginning of a line. The parser is required to evaluate <noteol> to the same string instance on both sides of the production.

```
<Tag>       ::= '_'{ <NonBlankChar>}+
<Value>     ::= { '.'|'?'|<Numeric>|
                  <CharString>|<TextField> }
<Numeric>   ::= { <Number>|
                  <Number> '(' <UnsignedInteger> ')' }
<Number>    ::= { <Integer>|<Float> }
<Integer>   ::= { '+'|'-' }? <UnsignedInteger>
<Exponent>  ::= { {'e'|'E' }|{'e'|'E' } { '+'|'- ' } }
                  <UnsignedInteger>
<UnsignedInteger> ::= {<Digit> }+
<Digit>     ::= {'0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'}
<Float>     ::= { <Integer><Exponent>|
                  { {'+'|'-'} ?
                    { {<Digit>}* '.' <UnsignedInteger>} |
                      {<Digit>}+ '.' }
                  }
                  {<Exponent>} ?
                }
               }
<CharString> ::= <UnquotedString>|<SingleQuotedString>|
     <DoubleQuotedString>
<eol><UnquotedString> ::=
     <eol><OrdinaryChar>{<NonBlankChar>}*
<noteol><UnquotedString> ::=
     <noteol>{<OrdinaryChar>|';'} {<NonBlankChar>}*
```

2.2.7.3.1. *CIF grammar*

(58) A CIF may be an empty file, or it may contain only comments or white space, or it may contain one or more data blocks. Comments before the first block are acceptable, and there must be white space between blocks.

```
<CIF> ::= <Comments>? <WhiteSpace>?
          { <DataBlock>
             { <WhiteSpace> <DataBlock> }*
             { <WhiteSpace> }?
          }?
```

(59) For a data block, there must be a data heading and zero or more data items or save frames.

```
<DataBlock> ::= <DataBlockHeading>
                  { <WhiteSpace>
                    { <DataItems> | <SaveFrame> }
                  }*
```

(60) A data-block heading consists of the five characters **data_** (case-insensitive) immediately followed by at least one non-blank character selected from the set of ordinary characters or the non-quote-mark, non-blank printable characters.

```
<DataBlockHeading> ::= <DATA_> { <NonBlankChar> }+
```

(61) For a save frame, there must be a save-frame heading, some data items and then the reserved word **save_**.

```
<SaveFrame> ::= <SaveFrameHeading>
                {<WhiteSpace> <DataItems>}+
                <WhiteSpace> <SAVE_>
```

(62) A save-frame heading consists of the five characters **save_** (case-insensitive) immediately followed by at least one non-blank character selected from the set of ordinary characters or the non-quote-mark, non-blank printable characters.

```
<SaveFrameHeading> ::= <SAVE_> { <NonBlankChar> }+
```

(63) Data come in two forms:

(i) A data-name tag separated from its associated value by a `<WhiteSpace>`.

(ii) Looped data. The number of values in the body must be a multiple of the number of tags in the header.

```
<DataItems>  ::= <Tag> <WhiteSpace> <Value> |
                 <LoopHeader> <LoopBody>
<LoopHeader> ::= <LOOP_> { <WhiteSpace> <Tag> }+
<LoopBody>   ::= <Value> { <WhiteSpace> <Value> }*
```

2.2.7.4. Common semantic features

2.2.7.4.1. *Introduction*

(1) The Crystallographic Information File (CIF) standard is an extensible mechanism for the archival and interchange of information in crystallography and related structural sciences. Ultimately CIF seeks to establish an ontology for machine-readable crystallographic information – that is, a collection of statements providing the relations between concepts and the logical rules for reasoning about them.

Essential components in the development of such an ontology are:

(*a*) the basic rules of grammar and syntax, described in Sections 2.2.7.1 to 2.2.7.3;

(*b*) a vocabulary of the tags or data names specifying particular objects;

(*c*) a taxonomy, or classification scheme relating the specified objects;

(*d*) descriptions of the attributes and relationships of individual and related objects.

In the CIF framework, the objects of discourse are described in so-called data dictionary files that provide the vocabulary and taxonomic elements. The dictionaries also contain information about the relationships and attributes of data items, and thus encapsulate most of the semantic content that is accessible to software. In practice, different dictionaries exist to service different domains of crystallography and a CIF that conforms to a specific dictionary must be interpreted in terms of the semantic information conveyed in that dictionary.

However, some common semantic features apply across all CIF applications, and the current document outlines the foundations upon which other dictionaries may build more elaborate taxonomies or informational models.

2.2.7.4.2. *Definition of terms*

(2) The definitions of Section 2.2.7.1.2 also hold for this part of the specification.

2.2.7.4.3. *Semantics of data items*

(3) While the STAR File syntax allows the identification and extraction of tags and associated values, the interpretation of the data thus extracted is application-dependent. In CIF applications, formal catalogues of standard data names and their associated attributes are maintained as external reference files called data dictionaries. These dictionary files share the same structure and syntax rules as data CIFs.

(4) At the current revision, two conventions (known as dictionary definition languages or DDLs) are supported for detailing the meaning and associated attributes of data names. These are known as DDL1 (Hall & Cook, 1995) and DDL2 (Westbrook & Hall, 1995), and they differ in the amount of detail they carry about data types, the relationships between specific data items and the large-scale classification of data items.

(5) While it may be formally possible to define the semantics of the data items in a given data file in both DDL1 and DDL2 data dictionaries, in practice different dictionaries are constructed to define the data names appropriate for particular crystallographic applications, and each such dictionary is written in DDL1 or DDL2 formalism according to which appears better able to describe the data model employed. There is thus in practice a bifurcation of CIF into two dialects according to the DDL used in composing the relevant dictionary file. However, the use of aliases may permit applications tuned to one dialect to import data constructed according to the other.

2.2.7.4.4. *Data-name semantics*

(6) Strictly, data names should be considered as void of semantic content – they are tags for locating associated values, and all information concerning the meaning of that value should be sought in an associated dictionary.

(7) However, it is customary to construct data names as a sequence of components elaborating the classification of the item within the logical structure of its associated dictionary. Hence a data name such as **_atom_site_fract_x** displays a hierarchical arrangement of components corresponding to membership of nested groupings of data elements. The choice of components readily indicates to a human reader that this data item refers to the fractional *x* coordinate of an atomic site within a crystal unit

cell, but it should be emphasized from a computer-programming viewpoint that this is coincidental; the attributes that constrain the value of this data item (and its relationship to others such as `_atom_site_fract_y` and `_atom_site_fract_z`) must be obtained from the dictionary and not otherwise inferred.

(8) *Comment:* In practice data names described in a DDL2 dictionary are constructed with a period character separating their specific function from the name of the category to which they have been assigned. In the absence of a dictionary file, this convention permits the inference that the data item with name `_atom_site.fract_x` will appear in the same looped list as other items with names beginning `_atom_site.`, and that all such items belong to the same category.

2.2.7.4.5. *Name space*

(9) The intention of the maintainers of public CIF dictionaries is to formulate a single authoritative set of data names for each CIF dialect (*i.e.* DDL1 and DDL2), thus facilitating the reliable archive and interchange of crystallographic data. However, it is also permissible for users to introduce local data names into a CIF. Two mechanisms exist to reduce the danger of collision of data names that are not incorporated into public dictionaries.

(10) The character string `[local]` (including the literal bracket characters) is *reserved* for local use. That is, no public dictionary will define a data name that includes this string. This allows experimentation with data items in a strictly local context, *i.e.* in cases where the CIF is not intended for interchange with any other user.

(11) Where CIFs including local data items are expected to enjoy a public circulation, authors may register a *reserved prefix* for their sole use. The registry is available on the web at http://www.iucr.org/iucr-top/cif/spec/reserved.html.

A reserved prefix, *e.g. foo*, must be used in the following ways:

(i) If the data file contains items defined in a DDL1 dictionary, the local data names assigned under the reserved prefix must contain it as their first component, *e.g.* `_foo_atom_site_my_item`.

(ii) If the data file contains items defined in a DDL2 dictionary, then the reserved prefix must be:

(*a*) the first component of data names in a category defined for local use, *e.g.* `_foo_my_category.my_item`.

(*b*) the first component following the period character in a data name describing a new item in a category already defined in a public dictionary, *e.g.* `_atom_site.foo_my_item`.

(12) There is no syntactic property identifying such a reserved prefix, so that software validating or otherwise handling such local data names must scan the entire registry and match registered prefixes against the indicated components of data names. Note that reserved prefixes may not themselves contain underscore characters.

2.2.7.4.6. *Note on handling of units*

(13) The published specification for CIF version 1.0 permitted data values expressed in different units to be tagged by variant data names (Hall *et al.*, 1991, p. 657):

> ... Many numeric fields contain data for which the units must be known. Each CIF data item has a default units code which is stated in the CIF Dictionary. If a data item is not stored in the default units, the units code is appended to the data name. For example, the default units for a crystal cell dimension are ångströms. If it is necessary to include this data item in a CIF with the units of picometres, the data name of `_cell_length_a` is replaced by `_cell_length_a_pm`. Only those units defined in the CIF Dictionary are acceptable. The default units, except for the ångström, conform to the SI Standard adopted by the IUCr.

This approach is deprecated and has not been supported by any official CIF dictionary published subsequent to version 1.0 of the core. All data values must be expressed in the single unit assigned in the associated dictionary.

A small number of archived CIFs exist with variant data names as permitted by the above clause. If it is necessary to validate them against versions of the core dictionary subsequent to version 1.0, the formal compatibility dictionary cif_compat.dic (ftp://ftp.iucr.org/cifdics/cif_compat.dic) may be used for the purpose. *No other use should be made of this dictionary.*

2.2.7.4.7. *Data-value semantics*

(14) The STAR syntax permits retrieval of data by simply requesting a specific data name within a specific data block. Prior knowledge about data type (*e.g.* text or numbers), whether the item is looped or whether the item exists in the file at all is unnecessary. However, applications in general need to know data type, valid ranges of values and relationships between data items, and a program designer needs to know the purpose of the data item (*i.e.* what physical quantity or internal book-keeping function it represents). While such semantic information may be defined informally for local data items (ones not intended for exchange between different users or software applications), formal descriptions of the semantics associated with data values are catalogued in data dictionary files. Currently two formalisms (dictionary definition languages) for describing data-value attributes are supported; full specifications of these formalisms (known as DDL1 and DDL2) are provided in Chapters 2.5 and 2.6.

2.2.7.4.7.1. *Data typing*

(15) Four base data types are supported in CIF. These are:

(i) **numb**: a value interpretable as a decimal base number and supplied as an integer, a floating-point number or in scientific notation;

(ii) **char**: a value to be interpreted as character or text data (where the value contains white-space characters, it must be quoted);

(iii) **uchar**: a value to be interpreted as character or text data but in a case-insensitive manner (*i.e.* the values `FOO` and `foo` are to be taken as identical);

(iv) **null**: a special data type associated with items for which no definite value may be stored in computer memory. It is the type associated with the special character literal values `?` (query mark) and `.` (full point), which may appear as values for any data item within a data file (see Section 2.2.7.4.8 below). It is also the type assigned to items defined in dictionary files that may not occur in data files.

(16) *Comment:* Many applications distinguish between multi-line text fields and character-string values that fit within a single line of text. While this is a convenient practical distinction for coding purposes, formally both manifestations should be regarded as having the same base type, which might be 'char' or 'uchar'. Applications are at liberty to choose whether to define specific multi-line text subtypes, and whether to permit casting between subtypes of a base type. The examples of character-string delimiters in Section 2.2.7.1.4(20) are predicated on an approach that handles all subtypes of character or text data equivalently.

(17) Where the attributes of a data value are not available in a dictionary listing, it may be assumed that a character string inter-

pretable as a number should be taken to represent an item of type 'numb'. However, an explicit dictionary declaration of type will override such an assumption.

2.2.7.4.7.2. *Subtyping*

(18) The base data types detailed in the previous section are very general and need to be refined for practical application. Refinement of types is to some extent application-dependent, and different subtypes are supported for data items defined by DDL1 and DDL2 dictionary files. The following notes indicate some considerations, but the relevant dictionary files and documentation should be consulted in each case.

(19) *DDL1 dictionaries*. Values of type 'numb' may include a standard uncertainty in the final digit(s) of the number where the associated item definition includes the attribute

```
_type_conditions    esd
```

(or `_type_conditions su`, a synonym introduced to DDL1 in 2005). For example, a value of 34.5(12) means 34.5 with a standard uncertainty of 1.2; it may also be expressed in scientific notation as 3.45E1(12).

(20) *DDL2 dictionaries*. DDL2 provides a number of tags that may be used in a dictionary file to specify subtypes for data items defined by that dictionary alone. Examples of the subtypes specified for the macromolecular CIF dictionary are:

code	identifying code strings or single words
ucode	identifying code strings or single words (case-insensitive)
uchar1	single-character codes (case-insensitive)
uchar3	three-character codes (case-insensitive)
line	character strings forming a single line of text
uline	character strings forming a single line of text (case-insensitive)
text	multi-line text
int	integers
float	floating-point real numbers
yyyy-mm-dd	dates
symop	symmetry operations
any	any type permitted

2.2.7.4.8. *Special generic values*

(21) The unquoted character literals `?` (query mark) and `.` (full point) are special and are valid expressions for any data type.

(22) The value `?` means that the actual value of a requested data item is *unknown*.

(23) The value `.` means that the actual value of a requested data item is *inapplicable*. This is most commonly used in a looped list where a data value is required for syntactic integrity.

2.2.7.4.9. *Embedded data semantics*

(24) The attributes of data items defined in CIF dictionaries serve to direct crystallographic applications in the retrieval, storage and validation of relevant data. In principle, a CIF might include as data items suitably encoded fields representing data suitable for manipulation by text processing, image, spreadsheet, database or other applications. It would be useful to have a formal mechanism allowing a CIF to invoke appropriate content handlers for such data fields; this is under investigation for the next CIF version specification.

2.2.7.4.10. *CIF conventions for special characters in text*

(25) The one existing example of embedded semantics is the text character markup introduced in the CIF version 1.0 specification and summarized in paragraphs (30)–(37) below. The specification is silent on which fields should be interpreted according to these markup conventions, but the published examples suggest that they may be used in any character field in a CIF data file except as prohibited by a dictionary directive. It is intended that the next CIF version specification shall formally declare where such markup may be used.

2.2.7.4.11. *Handling of long lines*

(26) The restriction in line length within CIF requires techniques to handle without semantic loss the content of lines of text exceeding the limit (2048 characters in this revision, 80 characters in the initial CIF specification). The line-folding protocol defined here provides a general mechanism for wrapping lines of text within CIFs to any extent within the overall line-length limit. A specific application where this would be useful is the conversion of lines longer than 80 characters to the CIF version 1.0 limit. This 80-character limit is used in the examples below for illustrative purposes.

These techniques are applied only to the contents of text fields and to comments.

In order to permit such folding, a special semantics is defined for use of the backslash. It is important to understand that this does not change the syntax of CIF version 1.0. All existing CIFs conforming to the CIF version 1.0 specification can be viewed as having exactly the same semantics as they now have. Use of these transformational semantics is optional, but recommended.

In order to avoid confusion between CIFs that have undergone these transformations and those that have not, the special comment beginning with a hash mark immediately followed by a backslash (`#\`) as the last non-blank characters on a line is reserved to mark the beginning of comments created by folding long-line comments, and the special text field beginning with the sequence line termination, semicolon, backslash (`<eol>;\`) as the only non-blank characters on a line is reserved to mark the beginning of text fields created by folding long-line text fields.

The backslash character is used to fold long lines in character strings and comments. Consider a comment which extends beyond column 80. In order to provide a comment with the same meaning which can be fitted into 80-character lines, prefix the comment with the special comment consisting of a hash mark followed by a backslash (`#\`) and the line terminator. Then on new lines take appropriate fragments of the original comment, beginning each fragment with a hash mark and ending all but the last fragment with a backslash. In doing this conversion, check for an original line that ends with a backslash followed only by blanks or tabs. To preserve that backslash in the conversion, add another backslash after it. If the next lexical token (not counting blanks or tabs) is another comment, to avoid fusing this comment with the next comment, be sure to insert a line with just a hash mark.

Similarly, for a character string that extends beyond column 80,

(i) first convert it to be a text field delimited by line termination–semicolon (`<eol>;`) sequences,

(ii) then change the initial line termination–semicolon (`<eol>;`) sequence to line termination–semicolon–backslash–line termination (`<eol>;\<eol>`),

(iii) and break all subsequent lines that do not fit within 80 columns with a trailing backslash. In the course of doing the translation,

(*a*) check for any original text lines that end with a backslash followed only by blanks or tabs;

(*b*) to preserve that backslash in the conversion, add another backslash after it, and then an empty line.

(More formally, the line folding should be done separately and directly on single-line non-semicolon-delimited character strings

to allow for recognition of the fact that no terminal line termination is intended – see below.)

In order to understand this scheme, suppose the CIF fragment (1) below were considered to have long lines. They could be transformed into (2) as follows:

(1) *Initial CIF*

```
#########################################################
### CIF submission form for Rietveld refinements
###
###                          Version 14 December 1998
###
#########################################################
data_znvodata
_chemical_name_systematic
; zinc dihydroxide divanadate dihydrate
;

_chemical_formula_moiety      'H2 O9 V2 Zn3, 2(H2 O)'
_chemical_formula_sum         'H6 O11 V2 Zn3'
_chemical_formula_weight      480.05
```

(2) *Transformed CIF*

```
#\
############################\
############################
### CIF submission form for Rietveld refinements
###
###                          Version 14 December 1998
###
#########################################################
data_znvodata
_chemical_name_systematic
;\
 zinc dihydroxide divan\
adate dihydrate
;

_chemical_formula_moiety
;\
H2 O9 V2 Zn3, 2(H2 O)\
;
_chemical_formula_sum         'H6 O11 V2 Zn3'
_chemical_formula_weight      480.05
```

In making the transformation from the backslash-folded form to long lines, it is very important to strip trailing blanks before attempting to recognize a backslash as the last character. When reassembling text-field lines, no reassembly should be done except in text fields that begin with the special sequence described above, line termination–semicolon–backslash–line termination, (`<eol>;\<eol>`), so that text fields that happen to contain backslashes but which were not created by folding long lines are not changed. It is also important to remove the trailing backslashes when reassembling long lines. The final line termination–semicolon sequence of a text field takes priority over the reassembly process and ends it, but a trailing backslash on the last line of a text field very nicely conveys the information that no trailing line termination is intended to be included within the character string.

Similarly, when reassembling long-line comments, the reassembly begins with a comment of the form hash–backslash–line termination. The initial hash mark is retained and then a forward scan is made through line terminations and blanks for the next comment, from which the initial hash mark is stripped and then the contents of the comment are appended. If that comment ends with a backslash, the trailing backslash is stripped and the process repeats. Note that the process will be ended by intervening tags, values, data blocks or other non-white-space information, and that the process will not start at all without the special hash–backslash–line termination comment.

Since there are very few, if any, CIFs that contain text fields and comments beginning this way, in most cases it is reasonable to adopt the policy of doing this processing unless it is disabled.

Here is another example of folding. The following three text fields would be equivalent:

```
;C:\foldername\filename
;

;\
C:\foldername\filename
;
```

and

```
;\
C:\foldername\file\
name
;
```

but the following example would be a two-line value where the first line had the value `C:\foldername\file\` and the second had the value `name`:

```
;
C:\foldername\file\
name
;
```

Note that backslashes should not be used to fold lines outside of comments and text fields. That would introduce extraneous characters into the CIF and violate the basic syntax rules. In any case, such action is not necessary.

2.2.7.4.12. *Dictionary compliance*

(27) Dictionary files containing the definitions and attribute sets for the data items contained in a CIF should be identified within the CIF by some or all of the data items

```
_audit_conform_dict_name
_audit_conform_dict_version
_audit_conform_dict_location
```

corresponding to DDL1 dictionaries or

```
_audit_conform.dict_name
_audit_conform.dict_version
_audit_conform.dict_location
```

for DDL2 dictionaries. Where no such information is provided, it may be assumed that the file should conform against the core CIF dictionary.

(28) The `_audit_conform` data items may be looped in cases where more than one dictionary is used to define the items in a CIF and they may include dictionaries of local data items provided such dictionary files have been prepared in accordance with the rules of the appropriate DDL.

(29) A detailed protocol exists for locating, merging and overlaying multiple dictionary files (McMahon *et al.*, 2000) (see Section 3.1.9).

2.2.7.4.13. *CIF markup conventions*

(30) If permitted by the relevant dictionary and if no other indication is present, the contents of a text or character field are assumed to be interpretable as text in English or some other human language. Certain special codes are used to indicate special characters or accented letters not available in the ASCII character set, as listed below.

2.2.7.4.14. *Greek letters*

(31) In general, the corresponding letter of the Latin alphabet, prefixed by a backslash character. The complete set is:

α	A	\a	\A	alpha	ν	N	\n	\N	nu
β	B	\b	\B	beta	o	O	\o	\O	omicron
χ	X	\c	\C	chi	π	Π	\p	\P	pi
δ	Δ	\d	\D	delta	θ	Θ	\q	\Q	theta
ε	E	\e	\E	epsilon	ρ	R	\r	\R	rho
φ	Φ	\f	\F	phi	σ	Σ	\s	\S	sigma
γ	Γ	\g	\G	gamma	τ	T	\t	\T	tau
η	H	\h	\H	eta	υ	U	\u	\U	upsilon
ι	I	\i	\I	iota	ω	Ω	\w	\W	omega
κ	K	\k	\K	kappa	ξ	Ξ	\x	\X	xi
λ	Λ	\l	\L	lambda	ψ	Ψ	\y	\Y	psi
μ	M	\m	\M	mu	ζ	Z	\z	\Z	zeta

2.2.7.4.15. *Accented letters*

(32) Accents should be indicated by using the following codes before the letter to be modified (*i.e.* use `\'e` for an acute e):

`\'`	acute (é)	`\"`	umlaut (ü)
`\=`	overbar or macron (ā)	`\``	grave (à)
`\~`	tilde (ñ)	`\.`	overdot (ȯ)
`\^`	circumflex (â)	`\;`	ogonek (ų)
`\<`	hacek or caron (ǒ)	`\,`	cedilla (ç)
`\>`	Hungarian umlaut or double accented (ő)	`\(`	breve (ŏ)

These codes will always be followed by an alphabetic character.

2.2.7.4.16. *Other characters*

(33) Other special alphabetic characters should be indicated as follows:

`\%a`	a-ring (å)	`\?i`	dotless i (ı)	`\&s`	German 'ss' (ß)
`\/o`	o-slash (ø)	`\/l`	Polish l (ł)	`\/d`	barred d (đ)

Capital letters may also be used in these codes, so an ångström symbol (Å) may be given as `\%A`.

(34) Superscripts and subscripts should be indicated by bracketing relevant characters with circumflex or tilde characters, thus:

superscripts	`Csp^3^`	for	Csp^3
subscripts	`U~eq~`	for	U_{eq}

The closing symbol is essential to return to normal text.

(35) Some other codes are accepted by convention. These are:

`\%`	degree (°)	`\\times`	×
`--`	dash	`+-`	±
`---`	single bond	`-+`	∓
`\\db`	double bond	`\\square`	□
`\\tb`	triple bond	`\\neq`	≠
`\\ddb`	delocalized double bond	`\\rangle`	>
`\\sim`	∼	`\\langle`	<
(*Note:* `~` is the code for subscript)		`\\rightarrow`	→
`\\simeq`	≃	`\\leftarrow`	←
`\\infty`	∞		

Note that `\\db`, `\\tb` and `\\ddb` should always be followed by a space, *e.g.* C=C is denoted by `C\\db C`.

2.2.7.4.17. *Typographic style codes*

(36) The codes indicated above are designed to refer to special characters not expressible within the CIF character set, and the initial specification did not permit markup for typographic style such as italic or bold-face type. However, in some cases the ability to indicate type style is useful, and in addition to the codes above HTML-like conventions are allowed of surrounding text by `<i> </i>` to indicate the beginning and end of italic, and by `<b> </b>` to indicate the beginning and end of bold-face type.

(37) If it is necessary to convey more complex typographic information than is permitted by these special character codes and conventions, the entire text field should be of a richer content type allowing detailed typographic markup.

References

Bernstein, H. J. (2002). *Some comments on parsing for computer programming languages.* http://www.bernstein-plus-sons.com/TMM/Parsing.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T. (1999). *Hypertext Transfer Protocol – HTTP/1.1.* RFC 2616. Network Working Group. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Freed, N. & Borenstein, N. (1996). *Multipurpose Internet Mail Extensions (MIME) Part two: media types.* RFC 2046. Network Working Group. http://www.ietf.org/rfc/rfc2046.txt.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

McMahon, B., Westbrook, J. D. & Bernstein, H. J. (2000). *Report of the COMCIFS Working Group on Dictionary Maintenance.* http://www.iucr.org/iucr-top/cif/spec/dictionaries/maintenance.html.

Ulrich, E. L. *et al.* (1998). XVIIth Intl Conf. Magn. Res. Biol. Systems. Tokyo, Japan.

Westbrook, J. D. & Hall, S. R. (1995). *A dictionary description language for macromolecular structure.* http://ndbserver.rutgers.edu/mmcif/ddl/.

2.3. Specification of the Crystallographic Binary File (CBF/imgCIF)

BY H. J. BERNSTEIN AND A. P. HAMMERSLEY

2.3.1. Introduction

The Crystallographic Binary File (CBF) format is a complementary format to the Crystallographic Information File (CIF) (Hall *et al.*, 1991) supporting efficient storage of large quantities of experimental data in a self-describing binary format. The image-supporting Crystallographic Information File (imgCIF) is an extension to CIF to assist in ASCII debugging and archiving of CBF files and to allow for convenient and standardized inclusion of images, such as maps, diagrams and molecular drawings, into CIFs for publication. The binary CBF format is useful for handling large images within laboratories and for interchange among collaborating groups. For smaller blocks of binary data, either format should be suitable. The ASCII imgCIF format is appropriate for interchange of smaller images and for long-term archiving.

CBF is designed to support efficient storage of raw experimental data (images) from area detectors with no loss of information, unlike some existing formats intended for this purpose. The format enables very efficient reading and writing of raw data, and encourages economical use of disk space. It may be coded easily and is portable across platforms. It is also flexible and extensible so that new data structures can be added without affecting the present definitions.

These goals are achieved by a simple file format, combining a CIF-like file header with compressed binary information. The file header consists of ASCII text giving information about the binary data as CIF tag–value pairs and tables. Each binary image is presented as a text-field value, either as raw octets of binary data in a CBF data set, or as an ASCII-based encoding of the same binary information in a true ASCII imgCIF data set. The ASCII-based encoded format uses e-mail MIME (Multipurpose Internet Mail Extensions) conventions to encode the binary data (Freed & Borenstein, 1996*a*,*b*,*c*; Freed *et al.*, 1996; Moore, 1996). The present version of the format tries to deal only with simple Cartesian data. These are essentially the 'raw' diffraction data that typically are stored in commercial formats or individual formats internal to particular institutes. Other forms of binary image data could be accommodated. It is hoped that CBF will replace individual laboratory or institute formats for 'home-built' detector systems, will be used as an inter-program data-exchange format, and will be offered as an output choice by commercial detector manufacturers specializing in X-ray and other detector systems. In this chapter we discuss the basic framework within which binary data and images are stored. The categories and data items that are used to describe beam and equipment axes, rastering methodologies, and image compression techniques are described in Chapter 3.7. The CBF/imgCIF dictionary is given in Chapter 4.6. An application programming interface (API) for the manipulation of image data is described in Chapter 5.6.

Affiliations: HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA; ANDREW P. HAMMERSLEY, ESRF/EMBL Grenoble, 6 rue Jules Horowitz, France.

2.3.2. CBF and imgCIF

CBF and imgCIF are two aspects of the same format. Since CIFs are pure ASCII text files, it was necessary to define a separate binary format to allow the combination of pseudo-ASCII sections and binary data sections. In the binary-file CBF format, the ASCII sections conform closely to the CIF standard but must use operating-system-independent 'line separators'. In order to facilitate interchange of files, an API that writes CBF files should use `\r\n` (carriage return, line feed) for the line separator. Use of this line separator allows the ASCII sections to be viewed with standard system utilities (*e.g.* 'more', 'pg') on a very wide range of operating systems (*e.g.* Unix, MacOS and Windows). However, an API that reads CBF format must accept any of the following three alternative line terminators as the end of an ASCII line: `\r`, `\n` or `\r\n`. As for all CIF data sets, an imgCIF file conforms to the normal text file-writing conventions of the system on which it is written. imgCIF is also one of the two names of the CIF dictionary (see Chapter 4.6) that contains the terms specific to describing image data in both CBF and imgCIF data sets. Thus a CBF or imgCIF data set uses data names from the CBF/imgCIF dictionary and other CIF dictionaries.

The general structure of a CBF or imgCIF data set is shown in Example 2.3.2.1. After a special comment to identify the file type (a so-called 'magic number') and any other initial comments, the data set begins with a '**`data_`*`blockname`***' which gives the name of the data block. Tags and values that describe the image data and how they were collected come next. For efficiency in processing, it is recommended that all the descriptive tags come before the actual image data. This recommendation is a *requirement* for the binary CBF format. It is *optional* for the ASCII imgCIF format. The image data are given as the value of the tag **`_array_data.data`**. The image data are given in a text field, using MIME conventions to describe the encoding.

2.3.2.1. A simple example

Before describing the format in full, we start by showing a simple but important and complete use of the format: that of storing a single detector image in a file together with a small amount of auxiliary information. This is intended to be a useful example that can be understood without reference to the full definitions. It also serves as an introduction or overview of the format definition. This example uses CIF DDL2-based dictionary items (see Chapter 2.6).

Example 2.3.2.2 relates to an image of 768 $\times$ 512 pixels stored as 16-bit unsigned integers, in little-endian byte order (this is the native byte ordering on a PC). The pixel sizes are 100.5 $\times$ 99.5 μm.

The example will be presented and discussed in three sections. The circled numerals (*e.g.* ①) are included to allow us to comment on portions of the example. They are not part of the CBF/imgCIF format.

The line marked by ①, starting with a hash character (**`#`**), is a CIF and CBF comment line. As a first line, the pattern of three hashes followed by 'CBF' helps to identify the data set as a CBF. It is a so-called 'magic number'. The text **`###CBF: VERSION`** must be present as the very first line of every CBF file. Following

Example 2.3.2.1. *General structure of a CBF or imgCIF data set. The critical values that define an image are marked with* ▌.

```
###CBF: VERSION 1.0
  data_blockname

  # cif tags and values describing the image, e.g.
  loop_
   _array_intensities.array_id
   _array_intensities.binary_id
   _array_intensities.linearity
   _array_intensities.undefined_value
   _array_intensities.overload
     image_1     1   linear     0      65535

# the image data are given as the value of
# the tag _array_data.data, usually in a loop_
# at the end of the data block.  The first two
# values identify the structure of the image
# and assure a unique identifier if there are
# multiple images of the same structure.

  loop_
   _array_data.array_id
   _array_data.binary_id
   _array_data.data

▌    image_1
     1

# The image itself begins and ends as a CIF
# text field, within which MIME conventions
# are used to describe the encoding of the image

▌;
▌--CIF-BINARY-FORMAT-SECTION--
▌Content-Type: application/octet-stream;
▌     conversions="x-CBF_PACKED"
▌Content-Transfer-Encoding: BINARY
▌X-Binary-Size: 374578
▌X-Binary-ID: 1
▌X-Binary-Element-Type:  "unsigned 16-bit integer"
▌Content-MD5: jGmkxkrpnizOetd9T/Np4NufAmA==

▌START_OF_BIN
▌*************<D5>9*****<D4>********* ...
▌[This is where the raw binary data would be – we can't print them here]
▌--CIF-BINARY-FORMAT-SECTION----
▌;
```

'VERSION' is the number of the corresponding version of the CBF/imgCIF extension dictionary and supporting documentation. Comment lines and white space (blanks and newlines) may appear anywhere outside the binary sections. In an imgCIF data set, the descriptive tags and values may be presented in any convenient order, *e.g.* the data could come first and the parameters necessary to interpret the data could come later. This order-independent convention holds for an imgCIF file, but for a CBF all the tags and values describing binary data (*i.e.* all the tags other than those in the ARRAY_DATA category) should be presented before the binary data, in the form of a header. This does not mean that there cannot be more useful information after the binary data. There could be another full header and more blocks of binary data. In the interest of efficiency in processing a CBF, the parameters that relate to a particular block of binary data must appear earlier in the CBF than the block itself.

The header begins at the line marked with ②. The `data_` token is the CIF token for identifying a data block. The name of the data block, `image_1`, follows immediately without any intervening white space. The name of the data block is arbitrary. Within a data block any given tag may be presented only once, either directly with a value following immediately, or as one of the column headings for the rows of a table. To reuse the same tag one must start a new data block.

Example 2.3.2.2. *A single image.*

```
###CBF: VERSION 1.0                                  ①
data_image_1                                         ②

_entry.id                                 'image_1'  ③
_chemical.entry_id                        'image_1'
_chemical.name_common                     'Protein X'

# Experimental details
_exptl_crystal.id                         'CX-1A'    ④
_exptl_crystal.colour                     'pale yellow'

_diffrn.id                                 DS1
_diffrn.crystal_id                        'CX-1A'

_diffrn_measurement.diffrn_id              DS1
_diffrn_measurement.method                 Oscillation
_diffrn_radiation_wavelength.id            L1
_diffrn_radiation_wavelength.wavelength
                                           0.7653
_diffrn_radiation_wavelength.wt            1.0

_diffrn_radiation.diffrn_id                DS1
_diffrn_radiation.wavelength_id            L1

_diffrn_source.diffrn_id                   DS1
_diffrn_source.source                      synchrotron
_diffrn_source.type                       'ESRF BM-14'

_diffrn_detector.diffrn_id                 DS1
_diffrn_detector.id                        ESRFCCD1
_diffrn_detector.detector                  CCD
_diffrn_detector.type              'ESRF Be XRII/CCD'

_diffrn_detector_element.id                1
_diffrn_detector_element.detector_id       ESRFCCD1

_diffrn_frame_data.id                      F1
_diffrn_frame_data.detector_element_id     1
_diffrn_frame_data.array_id                'image_1'
_diffrn_frame_data.binary_id               1
```

Information about the image begins at the line marked with ③. In the following lines, the apostrophes enclose strings that contain a space. Values that contain white space, or that could be confused with a CIF token, must always be quoted. A double quote (`"`) could have been used. There is a third way to quote a string, with the string `\r\n;`, *i.e.* with a semicolon at the beginning of a line, which allows multi-line strings to be presented. We shall use this form of text quotation for the binary data.

The experimental details begin at the line marked with ④. Many more data items could be defined, but here we are giving an example of one useful minimal (but not mandatory) set. See the imgCIF dictionary in Chapter 4.6 and the classification of image data in Chapter 3.7 for a discussion of which items are mandatory.

After describing the parameters of the experiment, we describe the organization of the image data (Example 2.3.2.3).

Note that we have changed from listing a value directly with each tag to a tabular format, using the CIF `loop_` token.

The `*.array_id` tags identify data items belonging to the same array. Here we have chosen the name `image_1`, but another name could have been used, as long as it is used consistently. The `*.index` tags refer to the dimension being defined, and the `*.dimension` column defines the number of elements in that dimension. The `*.precedence` tag defines the precedence of rastering of the data. In this case, the first dimension is the faster changing dimension. The `*.direction` column tells us the direction in which the data raster runs within a dimension. Here the data raster runs from the minimum element towards the maximum element ('increasing') in the first dimension, and from the maximum element towards the minimum element in the second dimension. This is the default rastering order.

Example 2.3.2.3. *Organization of image data in a CBF/imgCIF.*

```
# Define image storage mechanism
loop_
_array_structure.id
_array_structure.encoding_type
_array_structure.compression_type
_array_structure.byte_order
image_1  "unsigned 16-bit integer"  none
little_endian

loop_
_array_intensities.array_id
_array_intensities.binary_id
_array_intensities.linearity
_array_intensities.undefined_value
_array_intensities.overload
image_1     1   linear     0      65535

loop_
_array_structure_list.array_id
_array_structure_list.index
_array_structure_list.dimension
_array_structure_list.precedence
_array_structure_list.direction
image_1  1  768  1  increasing
image_1  2  512  2  decreasing

loop_
_array_element_size.array_id
_array_element_size.index
_array_element_size.size
image_1  1  100.5e-6
image_1  2  99.5e-6
```

Example 2.3.2.4. *Representation of the binary data.*

```
loop_
_array_data.array_id
_array_data.binary_id
_array_data.data

image_1 1
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-Size: 374578
X-Binary-ID: 1
X-Binary-Element-Type: "unsigned 16-bit integer"
Content-MD5: jGmkxkrpnizOetd9T/Np4NufAmA==

START_OF_BIN
************<D5>9******<D4>********* ...
[This is where the raw binary data would be – we can't print them here]
--CIF-BINARY-FORMAT-SECTION----
;
```

We have given the abstract ordering of the data. The physical view of data is described in detail in the CBF/imgCIF dictionary (Chapters 3.7 and 4.6).

In general, the physical sense of the image is from the sample to the detector.

The storage of the binary data is now fully defined. Further data items could be defined, but we are ready to present the image data (Example 2.3.2.4). This is done with the ARRAY_DATA category. The actual binary data will come just a little further down, as the essential part of the value of **_array_data.data**, which begins as semicolon-quoted text.

The line immediately after the line with the semicolon is a MIME boundary marker. As with all MIME boundary markers, it begins with '--' (two hyphens). The next few lines are MIME headers, describing some useful information we will need in order to process the binary section. MIME headers can appear in different orders and can be very confusing (look at the raw contents of an e-mail message with attachments), but there are only a few headers that have to be understood to process a CBF.

The 'Content-Type' header serves to describe the nature of the following data sufficiently to allow an association with an appropriate agent or mechanism for presenting the data to a user. It may be any of the discrete types permitted in RFC 2045 (Freed & Borenstein, 1996*b*), but, unless the binary data conform to an existing standard format (*e.g.* TIFF or JPEG), the description 'application/octet-stream' is recommended. If the octet stream has been compressed, the compression should be specified by the parameter **conversions="x-CBF_PACKED"** or by specifying one of the other compression types allowed as described in Chapter 5.6.

The 'Content-Transfer-Encoding' header describes any encoding scheme applied to the data, most commonly to transform it to an ASCII-only representation. For a CBF the value should be 'BINARY'. We consider the other values used for imgCIF below.

The 'X-Binary-Size' header specifies the size of the binary data in octets. Calculation of the size where compression is used is described in Section 2.3.3.3. The 'X-Binary-Element-Type' header specifies the type of binary data in the octets, using the same descriptive phrases as in **_array_structure.encoding_type** (the default value is 'unsigned 32-bit integer').

The other MIME headers in the example provide an identifier and a content checksum. The MIME header items are followed by an empty line and then by a special sequence (marked here as **START_OF_BIN**), consisting of the single characters Ctrl-L, Ctrl-Z, Ctrl-D and a single binary flag character of hexadecimal value D5 (213 decimal). The binary data follow immediately after this flag character. The reasons for choosing this sequence are discussed in Section 2.3.3.3.

After the last octet (*i.e.* byte) of the binary data, there is a special trailer `\r\n--CIF-BINARY-FORMAT-SECTION----\r\n;`. This repeats the initial boundary marker with an extra `--` at the end (a MIME convention for the last boundary marker), followed by a closing semicolon quote for a text section. This is essential in an imgCIF, and we include it in a CBF for consistency.

2.3.3. Overview of the format

This section describes the major elements of the CBF format.

(1) CBF is a binary file, containing self-describing array data, *e.g.* one or more images, and auxiliary data, *e.g.* describing the experiment.

(2) Apart from the handling of line terminators, the way binary data are presented and more liberal rules for ordering information, an ASCII imgCIF file is the same as a CBF binary file.

(3) A CBF consists of pseudo-ASCII text header sections consisting of 'lines' of no more than 80 ASCII characters separated by 'line separators', which are the pair of ASCII characters carriage return (`\r`, ASCII 13) and line feed (`\n`, ASCII 10), followed by zero, one or more binary sections presented as 'binary strings'. The file returns to the pseudo-ASCII format after each string, allowing additional binary strings to appear later in the file after additional headers.

(4) An imgCIF consists of ASCII lines of no more than 80 characters using the the normal line-termination conventions of the current system (*e.g.* `\n` in UNIX) with MIME-encoded binary strings at any appropriate point in the file. (For both CBF and imgCIF, the limitation of 80 characters per line will be increased to 2048 as CIF 1.1 is adopted.)

(5) The very start of the file has an identification item (magic number). This item also describes the CBF version or level (see Section 2.3.3.1 below).

(6) In the CBF binary format, the descriptive tags and values may be viewed as a 'header' section for the binary data section. The start of a header section is delimited by the usual CIF `data_` token (see Section 2.3.3.2 below).

(7) The header information must contain sufficient data names to fully describe the binary data sections.

(8) The binary data are presented as 'binary strings', which in CBF are specially formatted binary data within a semicolon-delimited text string. In imgCIF files, the binary data are MIME-encoded within true ASCII text fields.

(9) White space may be used within the pseudo-ASCII sections prior to the 'start of binary section' identifier to align the start-binary data sections to word or block boundaries. Similar use may be made of unused bytes within binary sections. However, no blank lines should be introduced among the MIME headers, since that would terminate processing of those headers and start the scan for binary data. In general, no guarantee is made of block or word alignment in a CBF of unknown origin.

(10) The end of the file need not be explicitly indicated, but including a comment of the form `###_END_OF_CBF` (including the carriage return, line feed pair) can help in debugging.

(11) For the most efficient processing of a CBF file, all binary data described in a single data block should appear as the last information in that data block. However, since binary strings can be parsed anywhere within the context of a CBF or imgCIF file, it is *recommended* that processing software for CBF accept such strings in any order, and it is *mandatory* that processing software for imgCIF accept such strings in any order. The binary identifier values used within a given data block section, and hence the binary data, must be unique for any given `*.array_id`, and it would be best to make them globally unique. However, a different data block may reuse binary identifier values. (This allows concatenation of files without renumbering the binary identifiers, and provides a certain level of localization of data within the file to avoid programs having to search potentially huge files for missing binary sections.)

(12) The recommended file extension for a CBF is 'cbf'. This allows users to recognize file types easily and gives programs a chance to 'know' the file type without having to prompt the user. However, a program should check for at least the file identifier to ensure that the file type is indeed CBF.

(13) The recommended file extensions for imgCIF are 'icf' or 'cif'.

(14) CBF format files are binary files, so when ftp is used to transfer files between different computer systems 'binary' or 'image' mode transfer should be selected.

(15) imgCIF files are ASCII files, so when ftp is used to transfer files between different computer systems 'ascii' transfer should be selected.

2.3.3.1. Details of the magic number

The magic number identifier is `###CBF: VERSION`. This must always be present so that a program can easily identify a CBF by simply inputting the first 15 characters. [The space is a blank (ASCII 32) and not a tab. All identifier characters must be upper case.] The first hash means that this line within the CIF is a comment line. Three hashes mean that this is a line describing the binary file layout for CBF. (All CBF internal identifiers start with three hashes, and all others must immediately follow a 'line separator'.) No white space may precede the first hash sign. Following the file identifier is the version number of the file; *e.g.* the full line might appear as `###CBF: VERSION 1.0`. The version number must be separated from the file identifier characters by white space, *e.g.* a blank (ASCII 32). The version number is defined as a major version number and minor version number separated by a decimal point. A change in the major version may mean that a program for the previous version cannot input the new version as some major change has occurred to CBF. A change in the minor version may also mean incompatibility if the CBF has been written using some new feature. For example, a new form of linearity scaling may be specified; this would be considered a minor version change. A file with the new feature would not be readable by a program supporting only an older version of the format.

2.3.3.2. Details of the header section

The start of a header section is delimited by the usual CIF `data_` token. Optionally, a header identifier, `###_START_OF_HEADER`, may be used before the `data_` token, followed by the carriage return, line feed pair, as an aid in debugging, but it is not required. (Naturally, another carriage return, line feed pair should immediately precede this and all other CBF identifiers, with the exception of the CBF file identifier at the very start of the file.) A header section, including the identification items which delimit it, uses only ASCII characters and is divided into 'lines'. The 'line separator' symbols `\r\n` (carriage return, line feed) are the same regardless of the operating system on which the file is written. This is an important difference from CIF, but must be so, as the file contains binary data and cannot be translated from one operating system to another, which is the case for ASCII text files. While a properly functioning CBF API should write the full `\r\n` line separator, it should recognize any of three sequences `\r`, `\n`, `\r\n` as valid line separators so that hand-edited headers will not be rejected.

The header section in a CBF obeys all CIF rules (Chapter 2.2) with the exception of the line separators, *i.e.*:

(i) 'Lines' are a maximum of 80 characters long. (This will be increased to 2048 in line with the CIF version 1.1 specification.)

(ii) All data names (tags) start with an underscore character '_'.

(iii) The hash symbol '`#`' (outside a quoted character string) means that all text up to the line separator is a comment.

(iv) White space outside character strings is not significant.

(v) Data names are case-insensitive.

(vi) The data item follows the data-name separator and may be of one of two types: character string (char) or number (numb). The type is specified for each data name.

(vii) Character strings may be delimited with single or double quotes, or blocks of text may be delimited by semicolons occurring as the first character on a line.

(viii) The `loop_` mechanism allows a data name to have multiple values. Immediately following the `loop_`, one or more data names are listed without their values, as column headings. Then one or more rows of values are given.

(ix) Any CIF data name may occur within the header section.

The tokens `data_` and `loop_` have special meaning in CIF and should not be used except in their indicated places. The tokens `save_`, `stop_` and `global_` also have special meaning in CIF's parent language, STAR, and should also not be used.

A single header section may contain one or more `data_` blocks.

2.3.3.3. Details of binary sections

All binary sections are contained within semicolon-delimited text fields, which may be thought of as 'binary strings'. Each such binary string must be given as a value of an appropriate CIF tag (usually **_array_data.data**), either directly or within a loop.

Before getting to the actual binary data, there are some preliminaries to allow a smooth transition from the conventions of CIF to those of raw streams of 'octets' (8-bit bytes). Within the binary-data text string, the conventions developed for transmitting e-mail messages including binary attachments are followed. There is secondary ASCII header information, formatted as MIME headers [see RFCs 2045–2049 by Freed & Borenstein (1996*a*,*b*,*c*), Freed *et al.* (1996) and Moore (1996)]. The boundary marker for the beginning of all this is the special string **--CIF-BINARY-FORMAT-SECTION--** at the beginning of a line. The initial '**--**' says that this is a MIME boundary. We cannot put '**###**' in front of it and conform to MIME conventions.

Immediately after the boundary marker are MIME headers, describing some useful information we will need to process the binary section. Only a few headers with a narrow range of values have to be understood to process a CBF (as opposed to an imgCIF, for which the headers can be more varied).

The 'Content-Type' header may be any of the discrete types permitted in RFC 2045 (Freed & Borenstein, 1996*b*); 'application/octet-stream' is recommended. If an octet stream was compressed, the compression should be specified by the parameter **conversions="x-CBF_PACKED"** or by the parameter **conversions="x-CBF_CANONICAL"**. Other compression schemes may be added at a future date. Details of the compression schemes are given in Chapter 5.6.

The 'Content-Transfer-Encoding' header should be **BINARY** for a CBF. For an ASCII imgCIF file, the 'Content-Transfer-Encoding' header may be **BASE64**, **QUOTED-PRINTABLE**, **X-BASE8**, **X-BASE10** or **X-BASE16**. The **BASE64** encoding is a reasonably efficient encoding of octets (approximately eight bits for every six bits of data) and is recommended for imgCIF files. The octal, decimal and hexadecimal transfer encodings are for convenience in debugging and are not recommended for archiving and data interchange.

The 'X-Binary-Size' header specifies the size of the binary data in octets. If compression was used, this size is the size after compression, including any book-keeping fields. In general, no portion of the binary header is included in the calculation of the size.

The 'X-Binary-Element-Type' header specifies the type of binary data in the octets, using the same descriptive phrases as in **_array_structure.encoding_type**. The full list of valid types is:

```
unsigned 8-bit integer
signed 8-bit integer
unsigned 16-bit integer
signed 16-bit integer
unsigned 32-bit integer
signed 32-bit integer
signed 32-bit real IEEE
signed 64-bit real IEEE
signed 32-bit complex IEEE
```

The default value is `unsigned 32-bit integer`. See IEEE (1985) for details on the IEEE formats.

The MIME header items are followed by an empty line and then by a special sequence marked here as **START_OF_BIN**, consisting of Ctrl-L, Ctrl-Z, Ctrl-D and a single binary flag character of hexadecimal value D5 (213 decimal). The binary data follow immediately after this flag character. The control (Ctrl-. . .) characters are inserted as a convenience to inhibit accidental listing or printing of large amounts of binary data on character-oriented output devices such as terminals or printers.

In general, if the value given for 'Content-Transfer-Encoding' is one of the real encodings **BASE64**, **QUOTED-PRINTABLE**, **X-BASE8**, **X-BASE10** or **X-BASE16**, this file is an imgCIF. More details on the imgCIF encodings are given in Section 2.3.5 below.

For either a CBF or an imgCIF, the optional 'Content-MD5' header provides a sophisticated check [the 'RSA Data Security, Inc. MD5 Message-Digest Algorithm' (Rivest, 1992)] on the integrity of the binary data.

In a CBF, the raw binary data begin after an empty line terminating the MIME headers and after the **START_OF_BIN** identifier. **START_OF_BIN** contains bytes to separate the ASCII lines from the binary data, bytes to try to stop the listing of the header, bytes which define the binary identifier, which should match the ***.binary_id** defined in the header, and bytes which define the length of the binary section.

Octet	Hexadecimal	Decimal	Purpose
1	0C	12 (Ctrl-L)	End the current page
2	1A	26 (Ctrl-Z)	Stop listings in MS-DOS
3	04	04 (Ctrl-D)	Stop listings in Unix
4	D5	213	Binary section begins
$5 \ldots 5+n-1$			Binary data (n octets)

Only bytes $5 \ldots 5+n-1$ are encoded for an imgCIF file using the indicated Content-Transfer-Encoding.

The binary characters serve specific purposes:

(i) The Ctrl-L will terminate the current page in listings in most operating systems.

(ii) The Ctrl-Z will stop the listing of the file in MS-DOS-type operating systems.

(iii) The Ctrl-D will stop the listing of the file in Unix-type operating systems.

(iv) The unsigned byte value 213 (decimal) is binary 11010101 (octal 325, hexadecimal D5). This has the eighth bit set, so it can be used to detect the error of seven-bit rather than eight-bit transmission. It is also asymmetric, providing one last check for reversal of bit order within bytes.

The carriage return, line feed pair before the **START_OF_BIN** and other lines can also be used to check that the file has not been corrupted (*e.g.* by being sent by ftp in ASCII mode).

The 'line separator' immediately precedes the 'start of binary identifier', but blank spaces may be added prior to the preceding 'line separator' if desired (*e.g.* to force word or block alignment). The binary data do not have to completely fill the bytes defined by the byte-length value, but clearly cannot be greater than this value (except when the value zero has been stored, which means that the size is unknown, and no other headers follow). The values of any unused bytes are undefined. The end of binary section identifier is placed exactly at the byte following the full binary section as defined by the length value. The end of binary section identifier consists of the carriage return/line feed pair followed by

```
--CIF-BINARY-FORMAT-SECTION----
;
```

with each of these lines followed by the carriage return/line feed pair. This brings us back into a normal CIF environment.

The first 'line separator' separates the binary data from the pseudo-ASCII line. The end-of-binary-section identifier is in a sense redundant, since the binary-data-length value tells a program how many bytes to jump over to the end of the binary data. However, this redundancy has been deliberately added for error checking, and for possible file recovery in the case of a corrupted file. This identifier must be present at the end of every block of binary data.

Example 2.3.4.1. *A multiple-block CBF with several images.*

```
###CBF: VERSION 1.0
# CBF file written by cbflib v0.6

# A comment cannot appear before the file identifier,
# but can appear anywhere else, except within the
# binary sections.

# Here the first data block starts
data_xxx

### ... various CIF tags and values here
###     but none that define array data items

# The "data_" identifier finishes the first data
# block and starts the second
data_yyy

### ... various CIF tags and values here including
###     ones that define array data items

loop_
_array_data.array_id
_array_data.binary_id
_array_data.data

image_1 1
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-Size: 3745758
X-Binary-ID: 1
X-Binary-Element-Type: "signed 32-bit integer"
Content-MD5: 1zsJjWPfol2GYl2V+QSXrw==

START_OF_BIN
<D5>^P<B8>P^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@ ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;
```

Example 2.3.4.1. *(cont.)*

```
# Following the "end of binary" identifier the file
# is pseudo-ASCII again, so comments are valid
# up to the next "start of binary" identifier.  Note
# that we have increased the binary ID by one.

image_1 2
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-Size: 3745758
X-Binary-ID: 2
X-Binary-Element-Type: "signed 32-bit integer"
Content-MD5: xR5kxiOetd9T/Nr5vMfAmA==

START_OF_BIN
<D5>^P<B8>P^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@ ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;

# Third binary section; note that we have a new
# array ID.

image_2 3
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-ID: 3
Content-MD5: yS5kxiOetd9T/NrqTLfAmA==

START_OF_BIN
*************'9*****`********* ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;

# Second header section
data_zzz

### ... various CIF tags and values here
###     including ones that define array
###     data items

# Since we only have one block left, we won't
# use a loop

_array_data.array_id   image
_array_data.binary_id  1
_array_data.data

# Note that we can put a comment here
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-ID: 1
Content-MD5: fooxiOetd9T/serNufAmA==

START_OF_BIN
*************'9*****`********* ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;

###_END_OF_CBF
```

2.3.4. A complex example

In Example 2.3.4.1 only the ARRAY_DATA category CIF identifiers are shown. The other CIF data items are not shown. This shows a possible structuring of a more complicated example. There are two header sections; the first contains two data blocks and defines three binary sections. CIF comment lines, starting with a hash mark '#', are used to clarify the structure.

2.3.5. imgCIF encodings

For an imgCIF, there are several alternative encodings for binary image data as ASCII text. Each binary image may use a different encoding in the same imgCIF data set or even in the same data block. The choice of encoding is specified in the 'Content-Transfer-Encoding' MIME header.

If the Transfer Encoding is `X-BASE8`, `X-BASE10` or `X-BASE16`, the data are presented as octal, decimal or hexadecimal data, respectively, organized into lines or words. Each word is created by composing octets of data in fixed groups of 2, 3, 4, 6 or 8 octets, either in the order . . .4321 ('big-endian') or 1234. . . ('little-endian'). If there are fewer than the specified number of octets to fill the last word, then the missing octets are presented as '`==`' for each missing octet. Exactly two equal signs are used for each missing octet even for octal and decimal encoding. The format of lines is

rnd xxxxxx xxxxxx xxxxxx

where *r* is H, O or D for hexadecimal, octal or decimal, *n* is the number of octets per word, and *d* is '<' or '>' for the '. . .4321' and '1234. . .' octet orderings, respectively. The '`==`' padding for the last word should be on the appropriate side to correspond to the missing octets, *e.g.*

```
H4< FFFFFFFF FFFFFFFF 07FFFFFF ====0000
```

or

```
H3> FF0700 00====
```

For these hexadecimal, octal and decimal formats only, comments beginning with '#' are permitted to improve readability.

BASE64 encoding follows MIME conventions. Octets are in groups of three, $c1$, $c2$, $c3$. The resulting 24 bits are reorganized in the following way (where we use the C operators $\gg$, $\ll$, & and | to denote, respectively, a right shift, a left shift, bit-wise intersection and bit-wise union). Four six-bit quantities are specified, starting with the high-order six bits $(c1 \gg 2)$ of the first octet, then the low-order two bits of the first octet followed by the high-order four bits of the second octet $((c1 \,\&\, 3) \ll 4 \,|\, (c2 \gg 4))$, then the bottom four bits of the second octet followed by the high-order two bits of the last octet $((c2 \,\&\, 15) \ll 2 \,|\, (c3 \gg 6))$, then the bottom six bits of the last octet $(c3 \,\&\, 63)$. Each of these four quantities is translated into an ASCII character using the mapping

```
          1         2         3         4
01234567890123456789012345678901234567890123456789
 | | | | | | |
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwx
5         6
01234567890123
 | | | | | | |
yz0123456789+/
```

Short groups of octets are padded on the right with one '=' if $c3$ is missing, and with '==' if both $c2$ and $c3$ are missing.

QUOTED-PRINTABLE encoding also follows MIME conventions, copying octets without translation if their ASCII values are 32...38, 42, 48...57, 59...60, 62, 64...126 and the octet is not a ';' in column 1. All other characters are translated to '=*nn*', where *nn* is the hexadecimal encoding of the octet. All lines are 'wrapped' with a terminating '=' (*i.e.* the MIME conventions for an implicit line terminator are never used).

Appendix 2.3.1
Deprecated CBF conventions

There was an earlier, now deprecated, CBF format in which the compression type was given as eight bytes of binary header. In this case, the eight bytes used for the compression type are subtracted from the size, so that the same size will be reported if the compression type is supplied in the MIME header. Use of the MIME header is the recommended way to supply the compression type.

These earlier versions of the specification also included three eight-byte words of information in binary that replicated information now available in the MIME header:

5...12	Binary section identifier (see **_array_data.binary_id**), 64-bit, little-endian	
13...20	The size (n) of the binary section in octets (*i.e.* the offset from octet 29 to the first byte following the data)	
21...28	Compression type:	
	CBF_NONE	0x0040 (64)
	CBF_CANONICAL	0x0050 (80)
	CBF_PACKED	0x0060 (96)
	...	

The three eight-byte words were followed by binary data. These words are not included when a MIME header is provided.

We are grateful to Frances C. Bernstein, Sydney R. Hall, Brian McMahon and Nick Spadaccini for their helpful comments and suggestions.

References

Freed, N. & Borenstein, N. (1996*a*). *Multipurpose Internet Mail Extensions (MIME) part five: Conformance criteria and examples.* RFC 2049. Network Working Group. http://www.ietf.org/rfc/rfc2049.txt.

Freed, N. & Borenstein, N. (1996*b*). *Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies.* RFC 2045. Network Working Group. http://www.ietf.org/rfc/rfc2045.txt.

Freed, N. & Borenstein, N. (1996*c*). *Multipurpose Internet Mail Extensions (MIME) part two: Media types.* RFC 2046. Network Working Group. http://www.ietf.org/rfc/rfc2046.txt.

Freed, N., Klensin, J. & Postel, J. (1996). *Multipurpose Internet Mail Extensions (MIME) part four: Registration procedures.* RFC 2048. Network Working Group. http://www.ietf.org/rfc/rfc2048.txt.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

IEEE (1985). *IEEE standard for binary floating-point arithmetic.* ANSI/IEEE Std 754–1985. New York: The Institute of Electrical and Electronics Engineers, Inc.

Moore, K. (1996). *MIME (Multipurpose Internet Mail Extensions) part three: Message header extensions for non-ASCII text.* RFC 2047. Network Working Group. http://www.ietf.org/rfc/rfc2047.txt.

Rivest, R. (1992). *The MD5 message-digest algorithm.* RFC 1321. Network Working Group. http://www.ietf.org/rfc/rfc1321.txt.

2.4. Specification of the Molecular Information File (MIF)

BY F. H. ALLEN, J. M. BARNARD, A. P. F. COOK AND S. R. HALL

2.4.1. Introduction

This volume is primarily concerned with methods for the exchange of crystallographic information, such as experimental conditions and measurements, computational procedures and results, and the geometrical description of three-dimensional (3D) chemical structures. Such information, now available for some 400 000 compounds (Allen & Glusker, 2002) is, of course, vitally important in chemistry and in many other branches of science. However, it must be appreciated that two-dimensional (2D) chemical structural diagrams are available for over eight million compounds, and are fundamental components of the language of chemistry at all levels. Two-dimensional graphical representations indicate atomic connectivities, formal bond types and residual atomic charges, and provide the universal formalism through which chemists communicate with each other on a daily basis and document their results.

In common with scientists in other disciplines, chemists were early users of computer technology. They solved their information needs through the creation of major databases of chemical compounds and the development of methods for searching these databases for complete structures or substructural fragments. From these data, software can compute the 3D structures and properties of molecules, and the resulting molecular images can be displayed and manipulated. Consequently, 2D chemical diagrams are at the heart of many computerized documentation systems, and are the basis for computational chemistry applications that form part of the routine armoury of the modern chemist.

Computationally, the 2D diagram is treated as a mathematical graph (Harary, 1972). The nodes of the graph represent atoms and the edges of the graph represent bonds. Each of these primary components can have additional attributes: element type, valency, charge *etc.* in the case of the atomic nodes, and bond type, cyclicity indicators *etc.* in the case of the bonded edges. Within this formalism, 2D display coordinates and 3D crystallographic or computed coordinates are additional atom attributes, while interatomic distances can further qualify the bonded edges. Using the concepts of graph theory, it is then possible to write algorithms for the analysis of chemical graphs, *e.g.* for the detection of chemical rings and ring systems (*e.g.* Wippke & Dyott, 1975), for the analysis of functional groups and their relationships *etc.* Most importantly, procedures have also been developed for the matching of complete chemical graphs (full graph isomorphism), and for the location of chemical substructures within a complete chemical graph (sub-graph isomorphism) (*e.g.* Feldmann *et al.*, 1977). In this way it is possible to achieve graphical substructure searches of very large collections of 2D chemical diagrams.

As with early crystallographic data exchange, structural chemistry applications use their own specialized formats for input, manipulation and output. The ready exchange of chemical data is often inhibited by specific data formats and by the enormous variation in methods used to represent 2D structures, stereochemical descriptors and certain 3D structural attributes. These are computational 'bottlenecks' that detract from an effective use of the large financial and intellectual investment in proprietary software and database systems. They have also contributed to the major need for in-house format conversion software, which must be continually upgraded and maintained to accommodate developmental changes within imported systems.

The need for a universal interchange format for chemical information became apparent in the late 1980s, at almost exactly the same time as crystallographers recognized a similar need. Data standards in structural chemistry involve many international organizations and individuals, and consequently a number of proposals for exchanging data were initially put forward. From these the Standard Molecular Data (SMD) format (Bebak *et al.*, 1989; Barnard, 1990) emerged as the leading contender. In the early 1990s, discussions between the SMD and CIF developers led to a re-expression of the SMD data items within the Self-defining Text Archive and Retrieval (STAR) File syntax (Hall, 1991; Hall & Spadaccini, 1994).

This chapter describes the initial core data definitions of a universal exchange format for chemistry, the Molecular Information File (MIF: Allen *et al.*, 1995), that arose from this coalescence of concepts and ideas. MIF is a complementary approach to CIF (Hall *et al.*, 1991). Because SMD was fundamental to the development of MIF, we begin this chapter with a brief history of this project.

2.4.2. Historical background

The Standard Molecular Data (SMD) format was initially developed by a group of European pharmaceutical companies in the mid-1980s. Draft documents were made available from 1987 and the specification was published (Bebak *et al.*, 1989). A meeting in Frankfurt in 1988 established a series of technical working groups under the auspices of the Chemical Structure Association (CSA) to examine the format specifications in detail and to make recommendations for any revision. As a result, a draft form of a revised format, described as SMD Version 5.0, was published in February 1990 (Barnard, 1990). A document describing the core format, *i.e.* those data items regarded as essential in any exchange file, was prepared by one of us (JMB) for consideration by Subcommittee E49.51 of the American Society for Testing and Materials (ASTM).

In December 1993, the ASTM subcommittee E49.51 approved a standard specification for the content (*i.e.* recommended data items) of computerized chemical structural files (ASTM, 1994), although the subcommittee did not publish any proposals for a format specification. Later, the *Chemical Abstracts* Service (CAS) circulated a draft proposal for a connection-table-based exchange format for chemical substances and queries. It used some ideas that are similar to the 1990 SMD proposal and is expressed within the framework of the Abstract Syntax Notation 1 (ISO, 2002*a*,*b*). MDL Information Systems Inc. has also published a description of their proprietary formats (Dalby *et al.*, 1992) and a number of other software systems now provide interfaces to these formats.

Affiliations: FRANK H. ALLEN, Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge, CB2 1EZ, England; JOHN M. BARNARD and ANTHONY P. F. COOK, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia.

During this period, the IUCr Working Party on Crystallographic Information had commissioned one of us (SRH) to coordinate the development of a universal file to replace the existing fixed-format Standard Crystallographic File Structure (SCFS: Brown, 1988). As documented in Chapter 1.1, the CIF approach was adopted as the international standard in 1990 and published by Hall *et al.* (1991). Although the small-molecule CIF is able to store a representation of 2D chemical topology, its data definitions do not meet all the needs of the chemical community. In 1991, the IUCr became interested in further extending CIF into the chemical arena and discussions took place between representatives of the CIF project and of the SMD Technical Working Group. These meetings decided that an integration of the SMD format and the STAR syntax was desirable because it provided a number of advantages over the existing SMD specifications (Barnard & Cook, 1992). In particular, SMD/STAR provides for a clearer separation of the data structure and the data content roles, together with more flexible data extensibility in future versions. In addition, automated data validation of STAR/SMD files is possible using electronic data dictionaries. In a wider context, there were obvious opportunities for integrating with other applications of the STAR File.

2.4.3. MIF objectives

Molecular information embraces a broad spectrum of data related to chemical and molecular structure. It includes both individual and linked data items, *inter alia* spectroscopic measurements, thermochemical data, electrochemical properties, crystal structure information and so on. These items represent the data descriptors of molecular chemistry and it is intended that all of these will eventually be accommodated in the MIF approach. However, the initial MIF implementation (Allen *et al.*, 1995), summarized in this chapter, treated only the important core information: the data items needed to specify the connectivity and stereochemistry of molecules and their 2D and 3D spatial representations. The MIF data items needed for more extensive applications must, in the future, involve the collaborative efforts of informatics and database experts from chemical industry and academia.

A dictionary of the initial MIF core data items described in this paper is given in Chapter 4.8. This is the abbreviated text version of the definition attributes contained in the electronic dictionary file. The core MIF data items provide descriptors for representing the 2D connectivity of a molecule or substructure, the conventions for relative or absolute stereochemical relationships, and the coordinates and conventions used for the generation of 2D and 3D graphical depictions. These data items apply to complete molecules, or to substructures with incomplete or variable attributes. As a consequence they are well suited for query definitions in substructure search systems, a feature that will be discussed later in this chapter.

2.4.4. MIF concepts and syntax

The syntax of the Molecular Information File is based on that of the STAR File (Hall, 1991; Hall & Spadaccini, 1994). A MIF is an ASCII text file that can be read or amended using a standard text editor, and that can be processed computationally without conversion to another format. The organization and expression of MIF data is summarized in Table 2.4.4.1. Each file consists of a series of data blocks and each block consists of a series of individual data items. There may be any number of items within a block and any number of blocks within a file. A data block represents a logical grouping of data items and, in most MIF applications, a data block will usually specify a complete chemical entity, *i.e.* a fully defined molecule or a query substructure.

Table 2.4.4.1. *Brief overview of the MIF syntax*

A text string is a string of characters bounded by white space, single or double quotes, or semicolons in column 1.
A data name is a text string bounded by white space starting with an underline.
A data value is a text string not starting with underline, preceded by an identifying data name.
A list is a sequence of data names, preceded by '`loop_`' and followed by a list of data values.
A save frame is a collection of data within a data block, preceded by '`save_`*`framecode`*' and closed with '`save_`'.
A data block is a collection of data, preceded by '`data_`*`blockcode`*'.
A global block is a collection of data, preceded by `global_`, that is common to all subsequent data blocks.
A file may contain any number of data blocks or global blocks.
A data name must be unique within a data block.

The MIF syntax, unlike that of a CIF, places no restrictions on line lengths or nested loop levels. For a detailed understanding of the differences between a MIF and a CIF, the reader should compare this chapter with Chapter 2.2 or refer to the published details of the STAR syntax (Hall & Spadaccini, 1994), the specification of the CIF core data items (Hall *et al.*, 1991) and the Dictionary Definition Language (Hall & Cook, 1995) used to define data items in the electronic version of a STAR dictionary.

CIF data, described by over a thousand items in the current dictionaries (see Part 4), encompass the fields of crystallographic structure and diffraction techniques, and these data items could readily be incorporated into a MIF. It should be noted, however, that currently the reverse is not possible because the current CIF syntax does not support nested loops or save frames.

2.4.4.1. Data identification

The fundamental principle that underpins MIF is exactly as for CIF: every data item is represented by a unique data tag followed by its associated data value. These combinations are referred to as tag–value pairs or tuples. Data names must start with an underscore (*i.e.* underline) character and data values may be any type of string, ranging from a single character to many lines of text. Here are some simple examples of MIF data items:

```
_atom_mass_number       79
_atom_type              Se
_display_colour         blue_medium
```

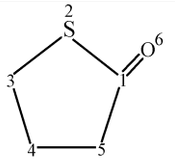
The complete list of MIF core data items is given in Chapter 4.8.

2.4.4.2. Looped lists

Repetitive data are stored in a MIF as lists of values, as they are in a CIF. Each list is prefaced by a `loop_` statement and a sequence of data names that identify the data values that follow in 'packets' of equal length. The values in each packet match the order and number of the data names. Any number of packets may appear in a looped list.

Atom and bond properties are typical of the information to appear in a looped list. The atoms and bonds of thiabutyrolactone in MIF format are shown in Fig. 2.4.4.1. The description of each data item in this example is given in Chapter 4.8, although the meanings are clear from the self-descriptive data names. The number of data values in each list is an exact multiple of the number of data names at the start of each loop structure. Looped lists are terminated by the next list or by any other data name, data block or end of file. Comments may be included in a MIF and are preceded by a `#` character, as illustrated in Fig. 2.4.4.1.

Hierarchical data may require the use of nested loop structures (see the `_display_*` loop in Fig. 2.4.4.2). Note that the packet for `_display_id` of 7 has two sets of `_display_conn_` values giving




```
# atom attributes list
     loop_
          _atom_id
          _atom_type
          _atom_attach_h
                    1  C  0     2  S  0     3  C  2
                    4  C  2     5  C  2     6  O  0

# bond attributes list
     loop_
          _bond_id_1
          _bond_id_2
          _bond_type_mif
                    1  2  S     2  3  S     3  4  S
                    4  5  S     5  1  S     1  6  D
```

Fig. 2.4.4.1. MIF coding of atom and bond properties for thiabutyrolactone.

connections to atom sites 1 and 4 (the other connections to site 7 appear in the next two packets). Data items that appear in looped lists are identified in the MIF dictionary (see Chapter 4.8) as having the attribute `_list` set to either 'yes' or 'both'. Other relationships between looped data items are also specified in the dictionary.

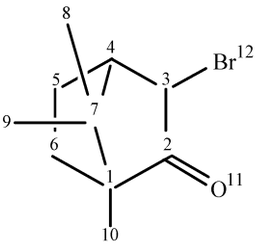
2.4.4.3. Save frames

Save frames are employed in a MIF to encapsulate grouped data for efficient cross-referencing. If a set of data needs to appear repeatedly in a data application, it is efficient to place this data into an addressable save frame. Molecular fragments, such as amino-acid units, are a case in point. A save frame is bounded by the statement **`save_`*framecode*** and terminated by a **`save_`** statement. It can be referenced within the parent data block using the value **`$`*framecode*** where the *framecode* matches the string in the **`save_`*framecode***. Note that all data names must be unique within a save frame, but the same data names may appear in other save frames or in the parent data block. Save frames may not contain other save frames but save-frame references (**`$`*framecode***) may appear in other save frames.

Save frames can be used in a MIF for many purposes. A simple application, the storage of alternative 3D conformational representations describing cyclohexane, is illustrated in Fig. 2.4.4.3. Within the STAR syntax, save-frame references (**`$`*framecode***) may occur before or after the save-frame definition within any data block. MIF preserves this basic STAR syntax. Save frames are particularly useful for defining commonly referenced structural templates and examples of this facility are discussed and illustrated (Figs. 2.4.7.1 and 2.4.7.2) in Section 2.4.7.

2.4.4.4. Data blocks

A data block is a sequence of unique data items or save frames. It is opened with a **`data_`*blockcode*** statement and closed by another data-block statement or a **`global_`** statement (see below). The *blockcode* string identifies the block within the file. Examples of data blocks are shown in Figs. 2.4.4.2, 2.4.4.3 and 2.4.6.1. Each data block in a file must have a unique *blockcode*.




```
data_bromocamphor

loop_
    _atom_id
    _atom_label
    _atom_type
    _atom_attach_h
    _atom_coord_x
    _atom_coord_y
    _atom_coord_z
          1  C1  C  0  4.69027   2.57756   2.10705
          2  C2  C  0  3.61112   2.44777   3.16754
          3  C3  C  1  4.16317   2.26258   0.69475
          4  C4  C  1  5.39943   1.87018  -0.13167
          5  C5  C  2  6.48959   2.15784   0.79703
          6  C6  C  2  6.57842   3.69178   1.11990
          7  C7  C  0  5.27609   3.93542   1.94650
          8  C8  C  3  5.93386   1.67891   2.14924
          9  C9  C  3  5.57194   0.17837   2.14326
          10 C10 C  3  6.85298   2.00528   3.35651
          11 O2  O  0  3.03484   2.36201   0.29318
          12 BR3 Br 0  5.41337   2.68686  -1.87368

loop_
    _bond_id_1
    _bond_id_2
    _bond_type_mif
          1  2  S   2  3  S   3  4  S   4  5  S
          5  6  S   6  1  S   1  7  S   7  8  S
          7  9  S   1 10  S   2 11  D   3 12  S
          4  7  S

_display_define_scale      50
_display_define_span_x    500
_display_define_span_y    500

loop_
    _display_id
    _display_object
    _display_symbol
    _display_colour
    _display_size
    _display_coord_x
    _display_coord_y
    loop_
        _display_conn_id
        _display_conn_symbol
        _display_conn_colour
  1     .  .       .  .  251 195 2 1b black stop_
  2     .  .       .  .  334 244 3 1b black stop_
  3     .  .       .  .  334 339 4 1b black stop_
  4     .  .       .  .  251 387 5 1b black stop_
  5     .  .       .  .  168 339 6 1b black stop_
  6     .  .       .  .  168 244 1 1b black stop_
  7     .  .       .  .  217 292 1 1b black
                                 4 1b black stop_
  8     .  .       .  .  191 426 7 1b black stop_
  9     .  .       .  .  100 292 7 1b black stop_
  10    .  .       .  .  251 100 1 1b black stop_
  11 text  O   blue 10   401 206 2 2b black stop_
  12 text Br yellow 10   417 387 1 1b black stop_
```

Fig. 2.4.4.2. MIF coding of atom properties (including 3D coordinates), bond properties and display information for (+)-3-bromocamphor.

```
data_cyclohexane

_molecule_name_common            cyclohexane

    loop_
        _atom_id
        _atom_type
        _atom_attach_h          1  C  2   2  C  2   3
                  C  2   4  C  2   5  C  2   6  C  2
loop_
    _bond_id_1
    _bond_id_2
    _bond_type_mif
           1 2 S  2 3 S  3 4 S  4 5 S  5 6 S  6 1 S

loop_
    _reference_conformation
                      $chair   $boat   $twisted_boat

save_chair
    loop_
        _atom_id
        _atom_coord_x
        _atom_coord_y
        _atom_coord_z    1   1.579  0.159  0.263
                         2   0.756  0.507 -0.986
                         3   0.825  0.493  1.541
                         4  -0.549 -0.131  1.590
                         5  -1.377  0.222  0.347
                         6  -0.626 -0.158 -0.937
save_

save_boat
    loop_
        _atom_id
        _atom_coord_x
        _atom_coord_y
        _atom_coord_z    1   1.657 -0.426  0.356
                         2   1.031  0.133 -0.927
                         3   0.960  0.133  1.602
                         4  -0.568 -0.040  1.558
                         5  -1.051 -0.738  0.279
                         6  -0.499 -0.028 -0.964
save_

save_twisted_boat
    loop_
        _atom_id
        _atom_coord_x
        _atom_coord_y
        _atom_coord_z    1   0.933  0.922  0.971
                         2   1.186  0.220 -0.368
                         3  -0.119  0.161  1.796
                         4  -1.135 -0.581  0.911
                         5  -1.371  0.181 -0.397
                         6  -0.083  0.236 -1.238
save_
```

Fig. 2.4.4.3. Atom and bond properties for cyclohexane, together with 3D coordinate representations of three alternative conformations: chair, boat and twisted boat.

2.4.4.5. Global blocks

A global block is similar to a data block except that it is opened with a **global_** statement and contains data that are common or 'default' to all subsequent data blocks in a file. Global data items remain active until re-specified in a subsequent data block or global block.

In some applications it may be efficient to place data that are common to all data blocks within a global block. In particular, save frames may be defined within global blocks and then referenced in subsequent data blocks [this statement corrects an error in Hall & Spadaccini (1994)]. Examples of global data are shown in Figs. 2.4.7.1 and 2.4.7.2, in which a variety of frequently referenced structural units are encapsulated within save frames specified in global blocks.

2.4.5. Atoms, bonds and molecular representations

The MIF dictionary (see Chapter 4.8) contains definitions of the principal data items needed to specify molecular connectivity and spatial representations. These definitions are grouped according to purpose or, as referred to in the DDL dictionary language (Hall & Cook, 1995), by category. Categories are formally specified in the MIF dictionary using the data attribute **_category** but they may also be identified from the data-name construction '***_<category>_<subcategory>_<descriptor>***'. Note that data items appearing in the same looped list must belong to the same category.

The values of some data items are restricted, by definition in the MIF dictionary, to standard codes or states. For example, the item **_bond_type_mif** can only have values S, D, T or O as in its dictionary definitions:

S: single (two-electron) bond;
D: double (four-electron) bond;
T: triple (six-electron) bond;
O: other (*e.g.* coordination) bond.

The MIF dictionary plays the important additional role of validating and standardizing data values. This is illustrated with the data item **_display_colour**, which identifies the colours of 'atom' and 'bond' graphical objects. The colour codes or states for this item are specified in its dictionary definitions as a set of permitted red/green/blue (RGB) ratios, and no other colours may be used in a MIF. This has the technical advantage of making colour states searchable for chemical applications.

Fig. 2.4.4.2 shows MIF data for the molecule (+)-3-bromocamphor. The 'atom' list contains the items **_atom_id**, **_atom_type** and **_atom_attach_h**, which identify the chemical properties of the atoms, plus the items **_atom_coord_x**, ***_y** and ***_z**, which specify the 3D molecular structure in Cartesian coordinates [these are taken from diffraction results (Allen & Rogers, 1970)]. The item **_atom_label** is also used with any graphical depiction of the 3D model. The 'bond' loop in this example uses the simple **_bond_type_mif** conventions described above. The data names needed to depict stereochemistry are discussed with examples (Figs. 2.4.8.1, 2.4.8.2 and 2.4.8.3) in Section 2.4.8.

The MIF approach to representing 2D chemical structure separates the specification of chemical atom and bond properties. This provides additional flexibility in the description of the graphical objects, such as atomic nodes and bonded connections. The MIF data required to generate a 2D chemical diagram are shown in Fig. 2.4.4.2. The diagram generated from this data will be in a display area of 500 × 500 coordinate units at a scale of 50 units per cm (the 2D chemical diagram shown in Fig. 2.4.4.2 is not to this scale). The default origin (the bottom left corner of the display area) can be specified with the item **_display_define_origin**. The data used to depict a 2D structure form a two-level loop with the 'atomic' graphical objects at level 1 and the 'bond' graphical objects at level 2. The item **_display_object** has the values '.' (null or no object), 'text' (an element or number string) or 'icon'. The size and colour of the atom site are specified with **_display_size** and **_display_colour**. The bonds connected to each atom site are specified as a sequence of **_display_conn_id** numbers (in loop level 2). These numbers must match one of the

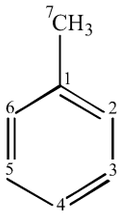
_display_id numbers at level 1. The connection object is specified with a **_display_conn_symbol** code, which must be a standard value in the dictionary definition, as is the colour of the icon if specified by **_display_conn_colour**.

2.4.6. Bonding conventions

Chemical information systems use a variety of conventions to specify attributes such as aromaticity, bond-order alternation, tautomerism *etc.* These system-dependent conventions decide the values that are permitted for quantities such as bond order, electronic charge and hydrogen-atom count. Most systems also provide for redundancy between chemical attributes. For example, the valency, the number of connected non-hydrogen atoms, the number of terminal hydrogen atoms and the bond types associated with a given atom are clearly related. Operational systems make use of these relationships to perform internal checks and to provide flexibility in substructure search processes.

The MIF data definitions provide for three bonding conventions. These are the data items **_bond_type_mif**, **_bond_type_casreg3** and **_bond_type_ccdc**. The 'mif' convention defines only single, double, triple and other bonds, while the 'casreg3' convention (Mockus & Stobaugh, 1980) extends these to include aromaticity in terms of 'ring alternating normalized bonds' and tautomerism *via* a 'tautomer normalized bond'. The 'ccdc' convention is that employed in the Cambridge Structural Database System (Allen *et al.*, 1991; Allen, 2002) to categorize bond types encountered in both organic and metal-organic molecules.

An important advantage of the MIF approach is that a molecule can be represented using all three bonding conventions within the same data block. An example of alternative bonding conventions encoded for toluene is shown in Fig. 2.4.6.1.




```
data_toluene

_molecule_name_common          toluene

loop_
    _atom_id
    _atom_type
    _atom_attach_h
          1  C  0    2  C  1    3  C  1    4  C  1
          5  C  1    6  C  1    7  C  3

loop_
    _bond_id_1
    _bond_id_2
    _bond_type_mif
    _bond_type_casreg3
    _bond_type_ccdc
          1  2  D  A  A    2  3  S  A  A
          3  4  D  A  A    4  5  S  A  A
          5  6  D  A  A    1  6  S  A  A
          1  7  S  S  S
```

Fig. 2.4.6.1. Three alternative bonding conventions for toluene stored in the same MIF data block.

2.4.7. Structural templates

In many chemical information systems, it is standard practice to build complete 2D molecular representations through the use of a library of commonly referenced structural templates, *e.g.* ligands, functional groups, amino-acid units *etc.*

In a MIF, molecular templates can be encapsulated as save frames, either within a data block for a specific molecule, or within a global block that is accessible to many data blocks. A simple application of a MIF template is shown in Fig. 2.4.7.1, where a 4-methylcyclohexyl ligand is used to encode the molecule tris(methylcyclohexyl)phosphine. In this example a molecular fragment is constructed in the save frame mechex, where the 'atom' sites and 'bond' connections appear in **_atom_*** and **_bond_*** loops. The molecule (2-methylcyclohexyl)(3-methylcyclohexyl)(4-methylcyclohexyl)phosphine is encoded by referencing the template fragment as the save frame $mechex. In the 'atom' loop, the item **_atom_environment** identifies the components of the target molecule as an 'atom' or 'frag' (fragment). If the component is a fragment, the items **_atom_frag_key** and **_atom_frag_id** are used to specify the frame code and the ID of the attached atom in the fragment, respectively. In the 'bond' loop, the connections from the atom P(1) to the template are encoded simply in terms of the **_atom_id** values. The necessary redefinition of the hydrogen and non-hydrogen counts of the template atoms is accomplished using the **_atom_attach_h** and **_atom_attach_nh** items, respectively. The external values override any values that are contained in, or derived from, the data in the template.

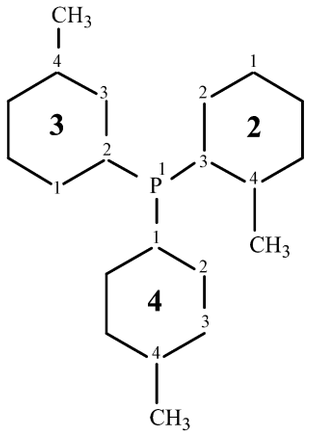
The same approach is used to construct the dipeptide alanylserine in Fig. 2.4.7.2. This employs the template peptide units described by the atoms and bonds in the save frames $alanyl and $seryl. The complete dipeptide is specified in its 'atom' list as the template peptides (identified by their save-frame names) and an additional carboxylate O atom. Note that only the atom sites affected by molecule formation are identified explicitly in this list, which gives the values of **_atom_attach_nh**, **_atom_attach_h** and **_atom_charge** for the modified sites in the zwitterionic form of alanylserine.

2.4.8. Stereochemistry and geometry at stereogenic centres

The Cahn–Ingold–Prelog (CIP) notation (Cahn *et al.*, 1966; Prelog & Helmchen, 1982) is available in the MIF definitions to specify the stereochemistry of a molecule. The CIP notation is restricted to tetrahedral atomic centres and to olefinic type stereogenic bonds, and the CIF approach is unsuitable for describing molecules with partially known stereochemistry, molecules containing more complex geometries or substructural queries. The MIF data items representing stereochemical quantities are as follows:

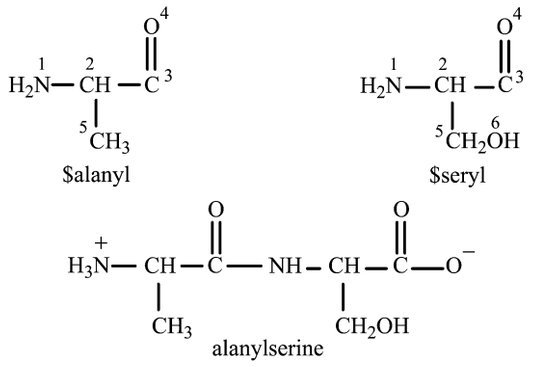
```
_define_stereo_relationship
_atom_cip
_bond_cip
_stereo_atom_id
_stereo_bond_id_1
_stereo_bond_id_2
_stereo_geometry
_stereo_vertex_id
```

The CIP stereochemical designators (R, S, E, Z, r, s, e, z *etc.*) are specified with the MIF data items **_atom_cip** and **_bond_cip**. The MIF atom-property data for the molecule (+)-3-bromocamphor are shown in Fig. 2.4.8.1. In this the absolute configuration is expressed as the atom CIP values *R*, *R* and *S* for nodes 1, 3 and 4. The period in this example is used to indicate a null field.



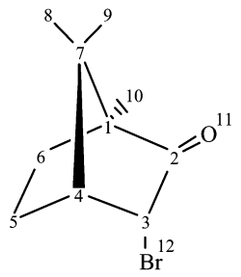
```
global_

save_mechex
     loop_
          _atom_id
          _atom_type
          _atom_attach_h
             1  C  2    2  C  2    3  C  2    4  C  1
             5  C  2    6  C  2    7  C  3
     loop_
          _bond_id_1
          _bond_id_2
          _bond_type_mif
             1  2  S    2  3  S    3  4  S    4  5  S
             5  6  S    1  6  S    4  7  S
save_

data_tris(methylcyclohexyl)phosphine

_molecule_name_common
; (2-methylcyclohexyl)(3-methylcyclohexyl)(4-
   methylcyclohexyl)phosphine
;

     loop_
          _atom_id
          _atom_environment
          _atom_frag_key
          _atom_frag_id
          _atom_type
          _atom_attach_nh
          _atom_attach_h
             1   atom   .         .  P  3  0
             2   frag   $mechex  3  C  3  1
             3   frag   $mechex  2  C  3  1
             4   frag   $mechex  1  C  3  1
     loop_
          _bond_id_1
          _bond_id_2
          _bond_type_mif
             1  2  S      1  3  S      1  4  S
```

Fig. 2.4.7.1. MIF representation of (2-methylcyclohexyl)(3-methylcyclohexyl)-(4-methylcyclohexyl)phosphine using a single global save frame that encapsulates the structure of methylcyclohexane, together with 'external' referencing of save-frame atoms in **_atom_** and **_bond_** loops.



```
global_

save_alanyl
     loop_
          _atom_id
          _atom_type
          _atom_attach_h
                1  N  2    2  C  1    3  C  0
                4  O  0    5  C  3
     loop_
          _bond_id_1
          _bond_id_2
          _bond_type_mif
                1  2  S    2  3  S    3  4  D
                2  5  S
save_
save_seryl
     loop_
          _atom_id
          _atom_type
          _atom_attach_h
                1  N  2    2  C  1    3  C  0
                4  O  0    5  C  2    6  O  1
     loop_
          _bond_id_1
          _bond_id_2
          _bond_type_mif
                1  2  S    2  3  S    3  4  D
                2  5  S    5  6  S
save_

data_alanylserine

_molecule_name_common     alanylserine
     loop_
          _atom_id
          _atom_environment
          _atom_frag_key
          _atom_frag_id
          _atom_type
          _atom_attach_nh
          _atom_attach_h
          _atom_charge
                1  frag   $alanyl   1   N   1   3  +1
                2  frag   $alanyl   3   C   3   0   0
                3  frag   $seryl    1   N   2   1   0
                4  frag   $seryl    3   C   3   0   0
                5  atom   .         .   O   1   0  -1
     loop_
          _bond_id_1
          _bond_id_2
          _bond_type_mif
                2 3 S    4 5 S
```

Fig. 2.4.7.2. MIF representation of the dipeptide alanylserine constructed using alanyl and seryl templates encapsulated in global save frames.




```
data_bromocamphor_2

_molecule_name_common       (+)-3-bromocamphor

_molecule_name_iupac
; 3R-bromo-1R,7,7-trimethyl-4S-bicyclo[2.2.1]heptan-
  2-one
;
    loop_
        _atom_id
        _atom_type
        _atom_attach_h
        _atom_cip
              1  C  0  R    2  C  0  .    3  C  1  R
              4  C  1  S    5  C  2  .    6  C  2  .
              7  C  0  .    8  C  3  .    9  C  3  .
             10  C  3  .   11  O  0  .   12  Br 0  .
```

Fig. 2.4.8.1. CIP stereochemical descriptors for (+)-3-bromocamphor.

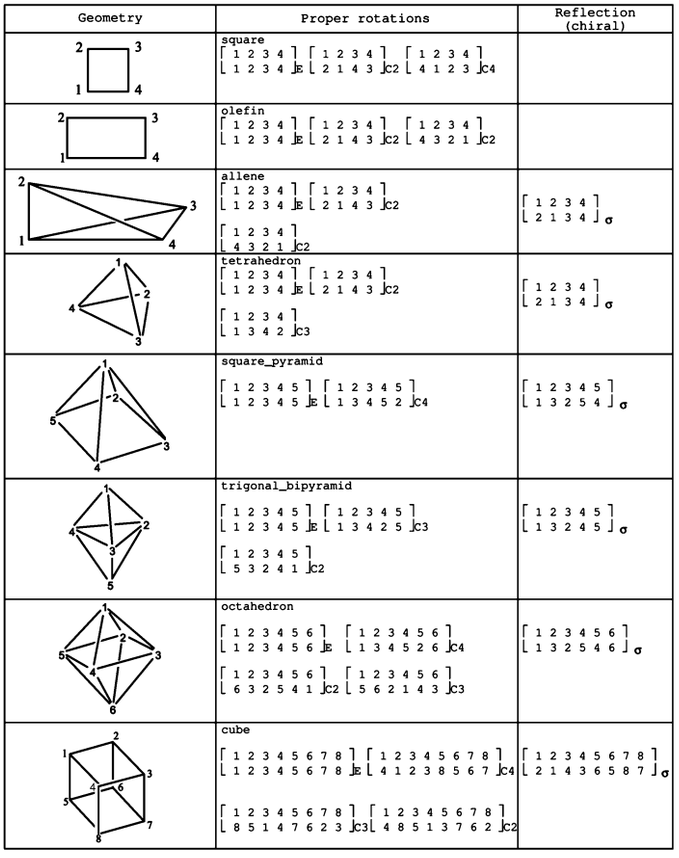


Fig. 2.4.8.2. Archetypal coordination geometries used in stereochemical definition of the MIF data item **_stereo_geometry**.

The stereogenic centre of a stereo group in a molecule has a relationship within that group that is specified by **_define_stereo_relationship**. Descriptions of the standard codes for **_define_stereo_relationship** are as follows.

absolute: The configuration of all stereogenic centres is exactly as described. This represents an enantiomerically pure compound with a known absolute configuration.

relative: The configuration of the stereogenic centres is only relative and the mirror reflection of the centres will also describe the same molecule. Only the configuration described in the MIF, or its mirror image, will be present in the molecule. This represents an enantiomerically pure compound with the described relative configuration.

racemic: The configuration of the stereogenic centres is only relative and the mirror reflection of the centres will also describe the same molecule. Both this configuration and its mirror image are present in a 1:1 ratio. This represents a racemic mixture of the molecule with the described relative configuration.

absolute_excess: The configuration of the stereogenic centres describes the absolute configuration of the excess component of a mixture of this configuration and its mirror reflection. This describes an enantiomeric excess in which the excess component has the described absolute configuration.

relative_excess: The configuration of the stereogenic centres is only relative. A mixture of this configuration and its mirror image is present, with one or other of the components in excess. This describes an enantiomeric excess mixture.

unknown: The configurational relationship between the stereogenic centres is not known.

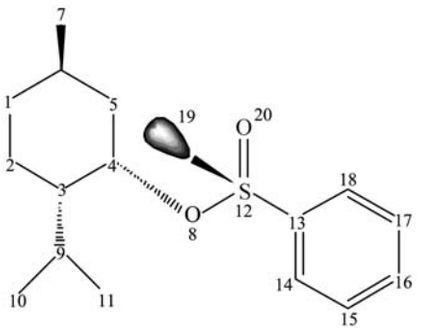
The geometry of each stereogenic centre is described individually in terms of a prototype geometrical model. The basic principles of this approach have been described elsewhere (Barnard *et al.*, 1990). The eight geometries currently defined for the MIF data item **_stereo_geometry** are given in Fig. 2.4.8.2. They include the organic stereogenic geometries (the tetrahedron, the rectangular description of olefin-related compounds and the anti-rectangle used to describe allene-related systems) and the common archetypal metal coordination geometries (square planar, tetrahedral, trigonal bipyramidal, square pyramidal, octahedral and cubic). This list is non-exclusive and can be extended as required in later versions of the MIF dictionary.

The vertex site of the geometrical model must be occupied by either an atom, an explicit or implicit hydrogen atom, or by an explicitly declared electron pair. In each case, there exist permutations of the enumerated vertices that, if applied, do not change the meaning of the description of the relevant stereo element. Thus, the MIF does not define a canonical ordering for citing geometric vertices and the comparison of two geometries requires the use of the permutation operators. These permutations are also indicated in Fig. 2.4.8.2.

For each stereogenic centre (defined by a **_stereo_atom_id**, or by **_stereo_bond_id_1** and ***_2**), the atom sites forming the stereochemical element specified by a **_stereo_geometry** code are stored as a sequence of **_stereo_vertex_id** values. An example of the specification of absolute stereochemistry, including the ordered enumeration of the tetrahedral vertices for the four stereogenic centres, is given in Fig. 2.4.8.3. In this example, the null symbol (a period) is used to indicate an implicit hydrogen atom or an unshared electron pair.

2.4.9. MIF query applications

A MIF is suitable for interrogating databases because data items are permitted to have a single value, or a 'sequence' of alternative values. This latter option is designated by the dictionary attribute **_type_conditions** which, for MIF applications, is set to

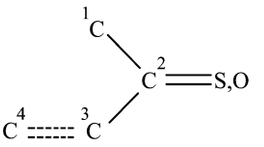


```
data_menthyl_p_toluenesulfinate

_molecule_name_common     menthyl-p-toluenesulfinate

_molecule_name_iupac
     "(1R,2S,5R)-(-)-menthyl (S)-p-toluenesulfinate"

loop_
    _atom_id
    _atom_type
    _atom_attach_h
    _atom_cip
          1 C 2 .    2 C 2 .    3 C 1 S    4 C 1 R
          5 C 2 .    6 C 1 R    7 C 3 .    8 O 0 .
          9 C 1 .   10 C 3 .   11 C 3 .   12 S 0 S
         13 C 0 .   14 C 1 .   15 C 1 .   16 C 1 .
         17 C 1 .   18 C 1 .   19 usp 0 . 20 O 0 .
loop_
    _bond_id_1
    _bond_id_2
    _bond_type_mif
          1  2 S  2  3 S  3  4 S  4  5 S  5  6 S
          6  7 S  4  8 S  3  9 S  9 10 S  9 11 S
          8 12 S 12 13 S 12 19 S 12 20 D 13 14 S
         14 15 D 15 16 S 16 17 D 17 18 S 13 18 D

_define_stereo_relationship             absolute

loop_
    _stereo_atom_id
    _stereo_geometry
    loop_
        _stereo_vertex_id
             6  tetrahedron    7  5  1  .  stop_
             3  tetrahedron    .  2  4  9  stop_
             4  tetrahedron    8  .  3  5  stop_
            12  tetrahedron   19 20 13  8  stop_
```

Fig. 2.4.8.3. Stereochemical data for menthyl-*p*-toluenesulfinate.

‘sequenced data’ (*via* the code `seq`). This permits a value string to contain alternative ‘values’ satisfying the following constructs: (*a*) the value string $v1, v2, v3$ signals that a data item must have the value $v1$ or $v2$ or $v3$, and (*b*) the value string $v1{:}v2$ signals that a data item must have a value in the range $v1$ to $v2$. Combinations of these constructions are permitted. All values must comply with the requirements defined by the attributes `_enumeration` and `_enumeration_range`.

An example of a substructural query in a MIF is shown in Fig. 2.4.9.1 for a conjugated ketone or thioketone fragment. Points of permitted variability of atom properties occur at atom 1, an sp^3 carbon atom that must have at least one attached hydrogen atom, and at atom 5, which can be S or O. The conjugated multiple C—C bond (3–4) is defined to be either localized double, delocalized double or aromatic using CCDC bonding conventions. Query coding of this type should be readily generated from most



```
data_query_example

_molecule_name_common
         "conjugated (thio)keto query substructure"

loop_
    _atom_id
    _atom_type
    _atom_attach_all
    _atom_attach_h
          1  C  4 1:3    2  C    3 O     3  C  3 .
          4  C  3 .      5  O,S  1 O

loop_
    _bond_id_1
    _bond_id_2
    _bond_type_ccdc
          1  2  S      2  3  S      2  5  D
          3  4  D,A,E
```

Fig. 2.4.9.1. Query substructure for conjugated ketones or thioketones. Atom C1 is sp^3 hybridized (total number of attached hydrogen and non-hydrogen atoms = 4) and carries at least one hydrogen atom. Bond C3—C4 may be localized double (D), aromatic (A) or delocalized double (E) in CCDC conventions.

graphical 2D search interfaces or be readable directly by a variety of 2D substructure search programs.

2.4.10. Conclusion

The present proliferation of formats for chemical applications tends to inhibit and complicate the exchange and use of chemical data. Many widely used chemical formats have a finite half-life because they are inflexible and not readily extensible. Others offer universality [*e.g.* Abstract Syntax Notation 1 (ISO, 2002*a*,*b*); Dalby *et al.* (1992); see http://www.daylight.com/smiles/] but lack visual simplicity, generality or machine readability. Nevertheless, the Molecular Information File approach has these properties but needs significantly more development to be a viable exchange approach for mainstream chemistry. The MIF dictionary enables chemical data items to be defined at high precision and this offers real benefits for the creation of a domain ontology in this field.

Herein we have outlined the basic MIF approach and provided definitions for an initial core of data items that are fundamental for the representation of 2D and 3D chemical structures and 2D substructures. These core data items cover most of the basic chemical data-exchange requirements of molecular modelling and database applications, but are clearly only a first step towards the level of chemical data exchange needed. Future MIF developments in applications software and, particularly, in data definitions are expected to encompass other aspects of chemistry. These developments will need the collaborative involvement and support of subject specialists from both academia and industry.

References

Allen, F. H. (2002). *The Cambridge Structural Database: a quarter of a million crystal structures and rising. Acta Cryst.* B**58**, 380–388.

Allen, F. H., Barnard, J. M., Cook, A. P. F. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Allen, F. H., Davies, J. E., Galloy, J. J., Johnson, O., Kennard, O., Macrae, C. F., Mitchell, E. M., Mitchell, G. F., Smith, J. M. & Watson, D. G. (1991). *The development of versions 3 and 4 of the Cambridge Structural Database System. J. Chem. Inf. Comput. Sci.* **31**, 187–204.

Allen, F. H. & Glusker, J. P. (2002). *Preface*. *Acta Cryst.* B**58**, Part 3.

Allen, F. H. & Rogers, D. (1970). *X-ray studies of terpenoid derivatives, Part III. A re-determination of the crystal structure of (+)-3-bromocamphor: the absolute configuration of (+)-camphor. J. Chem. Soc. B*, pp. 632–636.

ASTM (1994). *Standard specification for the content of computerized chemical structural information files or data sets.* ASTM Standard E 1586-93. American Society for Testing and Materials, Philadelphia, PA, USA.

Barnard, J. M. (1990). *Draft specification for revised version of the Standard Molecular Data (SMD) format. J. Chem. Inf. Comput. Sci.* **30**, 81–96.

Barnard, J. M. & Cook, A. P. F. (1992). *The Molecular Information File (MIF): a standard format for molecular information.* Report. Chemical Structure Association, London, England.

Barnard, J. M., Cook, A. P. F. & Rohde, B. (1990). *Beyond the structure diagram*, edited by D. Bawden & E. Mitchell, pp. 29–41. Chichester: Ellis Horwood.

Bebak, H., Buse, C., Donner, W. T., Hoever, P., Jacob, H., Klaus, H., Pesch, J., Roemelt, J., Schilling, P., Woost, B. & Zirz, C. (1989). *The Standard Molecular Data format (SMD format) as an integration tool in computer chemistry. J. Chem. Inf. Comput. Sci.* **29**, 1–5.

Brown, I. D. (1988). *Standard Crystallographic File Structure-87. Acta Cryst.* A**44**, 232.

Cahn, R. S., Ingold, C. K. & Prelog, V. (1966). *Specification of molecular chirality. Angew. Chem. Intl Ed. Engl.* **5**, 385–415.

Dalby, A., Nourse, J. G., Hounshell, W. D., Gushurst, A. K. I., Grier, D. L., Leland, B. A. & Laufer, J. (1992). *Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. J. Chem. Inf. Comput. Sci.* **32**, 244–255.

Feldmann, R. J., Milne, G. W. A., Heller, S. R., Fein, A., Miller, J. A. & Koch, B. (1977). *An interactive substructure search system. J. Chem. Inf. Comput. Sci.* **17**, 157–163.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR Dictionary Definition Language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

Harary, F. (1972). *Graph theory*, 3rd ed. London: Addison-Wesley.

ISO (2002*a*). ISO/IEC 8824-1. *Abstract Syntax Notation One (ASN.1). Specification of basic notation.* Geneva: International Organization for Standardization.

ISO (2002*b*). ISO/IEC 8825-1. *ASN.1 encoding rules. Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).* Geneva: International Organization for Standardization.

Mockus, J. & Stobaugh, R. E. (1980). *The Chemical Abstracts Service Chemical Registry System. VII. Tautomerism and alternating bonds. J. Chem. Inf. Comput. Sci.* **20**, 18–22.

Prelog, V. & Helmchen, G. (1982). *Basic principles of the CIP-system and proposals for a revision. Angew. Chem. Intl Ed. Engl.* **21**, 567–583.

Wippke, W. T. & Dyott, T. M. (1975). *Use of ring assemblies in ring perception algorithm. J. Chem. Inf. Comput. Sci.* **15**, 140–147.

2.5. Specification of the core CIF dictionary definition language (DDL1)

BY S. R. HALL AND A. P. F. COOK

2.5.1. Introduction

The CIF approach to data representation described in Chapter 2.2 is based on the STAR File universal data language (Hall, 1991; Hall & Spadaccini, 1994) detailed in Chapter 2.1. An important advantage of the CIF approach is the self-identification of data items through the use of tag–value tuples. This syntax removes the need for preordained data ordering in a file or stream of data and enables appropriate parsing tools to automate access independently of the data source. In this chapter, we will show that the CIF syntax also provides a higher level of abstraction for managing data storage and exchange – that of defining the meaning of data items (*i.e.* their properties and characteristics) as attribute descriptors linked to the identifying data tag.

Each attribute descriptor specifies a particular characteristic of a data item, and a collection of these attributes can be used to provide a unique definition of the data item. Moreover, placing the definitions of a selected set of data items into a CIF-like file provides an electronic dictionary for a particular subject area. In the modern parlance of knowledge management and the semantic web, such a data dictionary represents a *domain ontology*.

In most respects, data dictionaries serve a role similar to spoken-word dictionaries and as such are an important adjunct to the CIF data-management approach by providing semantic information that is necessary for automatic validation and compliance procedures. That is, prior lexical knowledge of the nature of individual items ensures that each item in a CIF can be read and interpreted correctly *via* the unique tag that is the link to descriptions in a data dictionary file. Because the descriptions in the dictionary are machine-parsable, the semantic information they contain forms an integral part of a data-handling process. In other words, machine-interpretable semantic knowledge embedded in data dictionaries leads directly to the automatic validation and manipulation of the relevant items stored in any CIF.

2.5.1.1. The concept of a dictionary definition language (DDL)

The structure or arrangement of data in a CIF is well understood and predictable because the CIF syntax may be specified succinctly (see Chapter 2.2 for CIF syntax expressed using extended Backus–Naur form). In contrast, the meaning of individual data values in a file is only known if the nature of these items is understood. For CIF data this critical link between the value and the meaning of an item is achieved using an electronic 'data dictionary' in which the definitions of relevant data items are catalogued according to their data name and expressed as attribute values, one set of attributes per item.

The dictionary definitions describe the characteristics of each item, such as data type, enumeration states and relationships between data. The more precise the definitions, the higher the level of semantic knowledge of defined items and the better the efficiency achievable in their exchange. To a large degree, the precision achievable hinges on the attributes selected for use in dictionary definitions, these being the semantic vocabulary of a dictionary. Within this context, attribute descriptors constitute a dictionary definition language (DDL). Definitions of the attributes described in this chapter are provided in Chapter 4.9.

The main purpose of this chapter is to describe the DDL attributes used to construct the core, powder, modulated structures and electron density CIF dictionaries detailed in Chapters 3.2–3.5 and 4.1–4.4. We shall see that each item definition in these dictionaries is constructed separately by appropriate choice from the attribute descriptors available, and that a sequence of definition blocks (one block per item) constitutes a CIF dictionary file. The organization of attributes and definition blocks in a dictionary file need not be related to the syntax of a data CIF, but in practice there are significant advantages if they are. Firstly, a common syntax for data and dictionary files enables the same software to be used to parse both. Secondly, and of equal importance, data descriptions and dictionaries need continual updating and additions, and the CIF syntax provides a high level of extensibility and flexibility, whereas most other formats do not. Finally, the use of a consistent syntax permits the dictionary attribute descriptors themselves to be described in their own DDL dictionary file.

While the basic functionality and flexibility of a dictionary is governed by the CIF syntax, the precision of the data definitions contained within it is determined entirely by the scope and number of the attribute descriptors representing the DDL. Indeed, the ability of a DDL to permit the simple and seamless evolution of data definitions and the scope of the DDL to precisely define data items both play absolutely pivotal roles *via* the supporting dictionaries in determining the power of the CIF data-exchange process.

2.5.2. The organization of a CIF dictionary

The precision and efficiency of a data definition language are directly related to the scope of the attribute vocabulary. In other words, the lexical richness of the DDL depends on the number and the specificity of the available language attributes. The breadth of these attributes, in terms of the number of separate data characteristics that can be specified, largely controls the precision of data definitions. However, it is the functionality of attributes that determines the information richness and enables higher-level definition complexity. For example, the attributes that define child–parent relationships between data and key pointers to items in list packets are essential to understanding the data hierarchy and to its validation. Attributes provide the semantic tools of a dictionary.

The choice and scope of attributes in the DDL are governed by both semantic and technical considerations. Attributes need to have a clear purpose to facilitate easy definition and comprehension, and their routine application in automatic validation processes. A CIF dictionary is much more than a list of unrelated data definitions. Each definition conforms to the CIF syntax, which requires each data block in the dictionary to be unique. However, the functionality of a dictionary involves elements of both relational and object-oriented processes. For example, attributes in one definition may refer to another definition *via* `_list_link_parent` or `_list_link_child` attributes, so as to indicate the dependency

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; ANTHONY P. F. COOK, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England.

of data items in lists. In this way the DDL, and consequently the dictionaries constructed from the DDL, invoke aspects of relational and object-oriented database paradigms. It is therefore useful to summarize these briefly here.

A relational database model (Kim, 1990) presents data as tables with a hierarchy of links that define the dependencies among tables. These explicit relationships enable certain properties of data to be shared, and, for related data values, to be derived. The structure of data links in a relational database is usually defined separately from the component data. This is an important strength of this approach. However, when data types and dependencies change continually, static relationships are inappropriate, and there is a need for non-relational extensions.

The object-oriented database model (Gray *et al.*, 1992) allows data items and tables to be defined without static data dependencies. A data item may be considered as a self-contained 'object' and its relationships to other objects handled by 'methods' or 'actions' defined within the objects. A database may have a base of statically defined explicit relationships with a dynamic layer of relationships provided by presenting some (or all) items as objects. Objects have well defined attributes, some of which may involve relationships with other data items, but objects need not have preordained links imposed by the static database structure.

The attributes and the functionality of CIF dictionaries incorporate aspects of both the basic relational and the object-oriented model. These provide the flexibility and extensibility associated with object-oriented data, as well as the relational links important to data validation and, ultimately, data derivation.

Table 2.5.4.1. *Comparison of DDL1 and DDL2 variants*

DDL1	DDL2
Data names identify the category of data as **_<category>_<detail>**	Data names identify the category as **_<category>.<detail>**
Definitions are declared as data blocks with **data_<dataname>**	Definitions are declared starting with **save_<dataname>** and ending with **save_**
An irreducible set of items is declared within one definition *e.g.* Miller indices h, k, l	All items are defined in separate frames related by **_item_dependent.name**
Items that appear in lists are identified with the attribute **list_**	List and non-list data items are not distinguished
List dependencies are declared within each definition *e.g.* **_list_reference**	Dependencies are declared in a category definition *e.g.* **_category_key.name**
	Identifies subcategories of data within category groupings *e.g.* matrix
	Provides aliases to equivalent names, including those in DDL1 dictionaries

2.5.3. Definition attributes

Efficient data exchange depends implicitly on the prior knowledge of the data. For CIF data, this knowledge is specified in a data dictionary using definition attributes. A unique set of attribute values exists for each kind of data item, be it numerical, textual or symbolic, because these characteristics represent its identity and function. This is illustrated below with two simple examples.

2.5.3.1. Example 1: attributes of temperature

Every numerical data item has distinct properties. Consider the number 20 as a measure of temperature in degrees. To understand this number it is essential to know its measurement units. If these are degrees Celsius, one knows the item is in the *temperature* class, *degrees Celsius* sub-class, and that a lower enumeration boundary of any value can be specified at -273.15. Such a constraint can be used in data validation. More to the point, without any knowledge of both the class and subclass, a numerical value has no meaning. The number 100 is unusable unless one knows what it is a measure of (*e.g.* temperature or intensity) and, equally, unless one knows what the units are (*e.g.* degrees Celsius, Kelvin or Fahrenheit; or electrons or volts).

2.5.3.2. Example 2: attributes of Miller indices

Knowing the inter-dependency of one data item on another plays a major role in the understanding and validation of data. If a triplet of numbers $5, 3, 0$ is identified as Miller indices h, k, l, one immediately appreciates the significance of the index triplet as a vector in reciprocal crystal space. This stipulates that the three numbers form a single non-scalar data item in which the indices are non-associative (*e.g.* $3, 5, 0$ is not equivalent to $5, 3, 0$) and irreducible (*e.g.* the index 3 alone has no meaning). As a reciprocal-space vector, the triplet has other properties if is part of a list of other reciprocal-space data items; namely, its value represents a key or list pointer (*i.e.* a unique key to a row of items in a list table) to other data items in the list associated with this vector. This means that data forming a 'reflection' list are inaccessible if these indices are absent, or invalid if there is more than one occurrence of the same triplet in the list. Such interdependency and relational information is very important to the application of data, and needs to be specified in a dictionary to enable unambiguous access and validation. Other types of data dependencies will be described in Section 2.5.6.

2.5.4. DDL versions

The capacity of a DDL to precisely define data items depends implicitly on the scope of the available attributes. It is quite possible, therefore, that a completely new data property cannot be specified using an existing DDL. In a field where data types evolve rapidly, the currently used dictionary language may be inadequate for the precise specification of an item. It is inevitable that future data items will exceed the capacity of existing dictionary attributes and methods to describe new data properties, and the dictionaries must evolve much in the same way that spoken languages continually adapt to changing modes of expression.

The first DDL used in crystallography (in 1990) was developed to compile a dictionary of 'core structural' crystallographic data items (Hall *et al.*, 1991). These data items were intended for use when submitting a manuscript to the journal *Acta Crystallographica* (and still are). The 'core' DDL is known commonly as DDL1 (Hall & Cook, 1995). Several years later, the definition of macromolecular crystallographic data items needed hierarchical descriptors for the different levels of structural entities, and an 'mmCIF' DDL, known as DDL2 (Westbrook & Hall, 1995), was developed. DDL2 was used to build the mmCIF dictionary (Bourne *et al.*, 1997). DDL1 is described in this chapter and DDL2 is described in Chapter 2.6. The DDL1 attributes have been used to construct the crystallographic core (fundamental structural), pd (powder diffraction), ms (modulated and composite structures) and rho (electron density) dictionaries. These dictionaries are discussed in Chapters

```
data_atom_site_Cartn_
    loop_ _name                 '_atom_site_Cartn_x'
                                '_atom_site_Cartn_y'
                                '_atom_site_Cartn_z'
    _category                    atom_site
    _type                        numb
    _type_conditions             esd
    _list                        yes
    _list_reference             '_atom_site_label'
    _units                       A
    _units_detail               'angstroms'
    _definition
;   The atom-site coordinates in angstroms specified
    according to a set of orthogonal Cartesian axes
    related to the cell axes as specified by the
    _atom_sites_Cartn_transform_axes description.
;
```

(*a*)

```
save__atom_site.Cartn_x
    _item_description.description
;   The x atom-site coordinate in angstroms
    specified according to a set of orthogonal
    Cartesian axes related to the cell axes as
    specified by the description given in
    _atom_sites.Cartn_transform_axes.
;
    _item.name                  '_atom_site.Cartn_x'
    _item.category_id            atom_site
    _item.mandatory_code         no
    _item_sub_category.id        cartesian_coordinate
    _item_aliases.alias_name    '_atom_site_Cartn_x'
    _item_aliases.dictionary     cif_core.dic
    _item_aliases.version        2.0.1
     loop_
    _item_dependent.dependent_name
                                 '_atom_site.Cartn_y'
                                 '_atom_site.Cartn_z'
    _item_related.related_name
                               '_atom_site.Cartn_x_esd'
    _item_related.function_code  associated_esd
    _item_type.code              float
    _item_type_conditions.code   esd
    _item_units.code             angstroms
     save_
```

(*b*)

Fig. 2.5.4.1. Comparison of DDL versions: (*a*) DDL1, (*b*) DDL2.

3.2–3.5 and presented in full in Chapters 4.1–4.4, respectively, of this volume.

It is helpful to give a brief comparison of the definition capabilities of DDL1 and DDL2. The differences may be best illustrated by simply comparing their application to the definition of the same data item. The two separate definitions of `_atom_site_Cartn_x` are shown in Fig. 2.5.4.1.

Although both DDL versions conform to the syntax of the STAR File (DDL1 conforms to the CIF syntax but DDL2 uses STAR File save frames not permitted in CIF data files) and are composed of similar attributes, there are clear differences. These are summarized in Table 2.5.4.1.

The similarity in the two DDL versions is such that software exists for parsing and validating CIFs against the DDL1 or DDL2 dictionaries interchangeably (*e.g.* see the CIF toolbox software *CIFtbx* described in Chapter 5.4).

2.5.5. The structure of DDL1 definitions

The organization of definitions in a CIF dictionary is straightforward. We are all familiar with how words are described in a

```
data_cell_formula_units_Z
    _name                       '_cell_formula_units_Z'
    _category                    cell
    _type                        numb
    _enumeration_range           1:
    _definition
; The number of the formula units in the unit cell
  as specified by _chemical_formula_structural,
  _chemical_formula_moiety or _chemical_formula_sum.
;
```

Fig. 2.5.5.1. DDL1 definition with a few attributes.

```
data_on_this_dictionary
    _dictionary_name            cif_example.dic
    _dictionary_version         0.0
    _dictionary_update          1999-03-15
    _dictionary_history
;  1999-03-11  Created as a dictionary example
   1999-03-15  Further simplifications
;
data_parameter_ABC
    _name                       '_parameter_ABC'
#        <<<<<<<<<< other data attributes here
data_factor_XYZ
    _name                       '_factor_XYZ'
#        <<<<<<<<<< other data attributes here
data_and_so_on
    _name                       '_and_so_on'
#        <<<<<<<<<< other data attributes here
```

Fig. 2.5.5.2. DDL1 definition showing basic dictionary organization.

spoken-language dictionary. Each defined word is followed by a sequence of attribute descriptions, such as phonetic annotation, grammatical context, word origins, meanings, examples of use and so on. The definition of data items in a CIF dictionary is organized in the same way. Each item description is preceded by the item's unique identifying tag as a datablock code, and is composed of a sequence of attribute items specifying the item's characteristics. A full description of the DDL1 attributes is given in Chapter 4.9.

A series of example definitions will be used to introduce the basic structure of the dictionary and definitions. The definitions are of familiar crystallographic data items, extracted from the core CIF dictionary in Chapter 4.1. Note that, in some cases, the text part of the definitions has been abbreviated for conciseness.

2.5.5.1. Definition example 1: formula units per cell (*Z*)

A definition for formula units in a crystal unit cell (*Z*) is shown in Fig. 2.5.5.1. The definition includes only the attributes `_name`, `_category`, `_type`, `_enumeration_range` and `_definition`. These identify the item's unique properties and enable its validation. The `_category` attribute indicates the class of the defined data item. Since lists may contain only items of one category type, this attribute is critical to data items appearing in lists (see Section 2.5.5.5). The attribute `_enumeration_range` has the value '`1:`' which stipulates that the item must be 'one or a higher number'.

2.5.5.2. Definition example 2: dictionary audit information

A CIF dictionary file contains, in addition to the data definitions, information about the nature of the dictionary. Fig. 2.5.5.2 shows the typical organization of definitions in a dictionary. Each dictionary starts with the audit information giving its version and creation history. This is followed by definitions in separate data

```
data_cell_length_
    loop_ _name                 '_cell_length_a'
                                '_cell_length_b'
                                '_cell_length_c'
    _category                    cell
    _type                        numb
    _type_conditions             esd
    _enumeration_range           0.0:
    _units                       A
    _units_detail               'angstroms'
    _definition
;   Unit-cell lengths in angstroms corresponding to
    the structure reported.
;
```

Fig. 2.5.5.3. DDL1 definition of irreducible data items.

```
data_atom_site_attached_hydrogens
_name             '_atom_site_attached_hydrogens'
    _category              atom_site
    _type                  numb
    _list                  yes
    _list_reference        '_atom_site_label'
    _enumeration_range     0:8
    _enumeration_default   0
    _definition
;   The number of hydrogen atoms attached to the atom
    at this site excluding any hydrogen atoms for
    which coordinates (measured or calculated) are
    given.
;
```

Fig. 2.5.5.4. DDL1 definition of a 'list' data item.

blocks. The name of a data block matches the defined data name or the initial portion thereof. Each definition data block contains a sequence of data declarations, one for each attribute. The attributes within a data block constitute the total definition information, and only those attributes appropriate to a given defined item need be specified.

2.5.5.3. Definition example 3: irreducible data items

Some data items are closely related to other data items. Fig. 2.5.5.3 shows the definition of the length components **_cell_length_a**, ***_b** and ***_c** of the crystal unit-cell metric tensor. Because of their different enumeration and units attributes, the definition blocks for cell lengths and angles are specified separately. This definition shows how data items closely related in function may be defined in the same data block. The **loop_** command is used to list the multiple **_name** values. The attribute **_type_conditions** is used to indicate that these values are measurements and can have a standard uncertainty ('s.u.') value appended in parentheses *e.g.* 7.254(2). (The label 'esd' reflects the historical but inaccurate terminology 'estimated standard deviation'.) The **_units** and **_units_detail** attributes stipulate that the measurement unit of length is ångströms.

2.5.5.4. Definition example 4: list data

The definition of the number of hydrogen atoms attached to an atom site is shown in Fig. 2.5.5.4. The attributes **_list** and **_list_reference** signal whether the item is used in a list and what key item is in that list. As described in Chapters 2.1 and 2.2, a CIF list is a two-dimensional table where data items are in the columns (headed by the data names) and the values are the rows. The attribute **_list** is `yes` if a data item may be used in a list. The attribute **_list_reference** specifies the key item as **_atom_site_label**. Each value of a key item must be unique so that the row of associated items can be accessed unambiguously. The attribute **_enumeration_default** specifies a default value for the defined item if it is absent from the data file.

2.5.5.5. Definition example 5: category information

Each CIF dictionary also contains example applications of defined data items, grouped according to category. Fig. 2.5.5.5 shows these for the category ATOM_SITE, as defined in the core dictionary (see Chapter 3.2). A **_type** attribute with a value of `null` specifies that a data block contains no definition information. The particular examples shown here illustrate that the reference key for the ATOM_SITE category is **_atom_site_label**. This data item is present in both examples shown, and in each packet the label is unique.

2.5.5.6. Definition example 6: mandatory and linked items

The example definition in Fig. 2.5.5.6 shows the attributes for the item **_atom_site_label**. As illustrated in Section 2.5.5.5, this item is the reference key to a list of items belonging to the category ATOM_SITE. The attribute **_list_mandatory**, which is set to a value of `yes`, specifies that this item is a mandatory item to a list of category ATOM_SITE items. The attribute **_list_link_child** identifies items in other categories that are 'linked' by derivation to this item and therefore share the same data values. This dependency is known as a child dependency. The definition shows that data items describing the labels of atom sites in a list of angular geometry, *i.e.* **_geom_angle_atom_site_label_1** and **_geom_angle_atom_site_label_2**, are the same labels described by **_atom_site_label** in the atom-site list. This is because the geometry is derived directly from the atom-site data. Note that this dependency requires that the ATOM_SITE list be present in the same CIF as the GEOM_ANGLE list, otherwise the molecular geometry information cannot be linked to the three-dimensional structural information.

2.5.5.7. Definition example 7: joinable lists

In the example definition shown in Fig. 2.5.5.7, the key item **_atom_site_aniso_label** is shown to have a special relationship with the key item **_atom_site_label**. This is because the data items in the list category ATOM_SITE_ANISO may be merged with (*i.e.* joined to) items in the list category ATOM_SITE. If this happens, the item **_atom_site_label** assumes the role as the key to the merged packets and this is signalled using the attribute **_list_link_parent**. When these two categories appear in separate lists, the category ATOM_SITE_ANISO data require the category ATOM_SITE data to be present in the data instantiation, but not *vice versa*. Note that the parent relationship, unlike that in the example of Section 2.5.5.6 arising from a derivation dependency, is because ATOM_SITE_ANISO is a subcategory of ATOM_SITE.

2.5.5.8. Definition example 8: equivalent items

As with the preceding example definition, this example concerns anisotropic atomic displacement parameters. Fig. 2.5.5.8 shows the definition of the component items **_atom_site_aniso_U_** that comprise an irreducible set of matrix elements. The key item in a list of ATOM_SITE_ANISO items is **_atom_site_aniso_label**. This definition illustrates how the attributes **_related_item** and **_related_function** are used to identify the similar data items **_atom_site_aniso_B_** that are atomic displacement parameters related by a simple conversion factor. Other attributes used to specify data relationships are described in Section 2.5.6 below.

```
################
## ATOM_SITE ##
################
data_atom_site_[]
    _name                       '_atom_site_[]'
    _category                     category_overview
    _type                         null
    loop_ _example
          _example_detail
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   Example 1 - based on data set TOZ of Willis,
    Beckwith & Tozer [Acta Cryst. (1991), C47,
    2276-2277].
;
;   loop_
    _atom_site_label
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
    _atom_site_U_iso_or_equiv
    _atom_site_adp_type
    _atom_site_calc_flag
    _atom_site_calc_attached_atom
O1   .4154(4)  .5699(1)  .3026(0)  .060(1)  Uani  ? ?
C2   .5630(5)  .5087(2)  .3246(1)  .060(2)  Uani  ? ?
C3   .5350(5)  .4920(2)  .3997(1)  .048(1)  Uani  ? ?
N4   .3570(3)  .5558(1)  .4167(0)  .039(1)  Uani  ? ?
# - - - - data truncated for brevity - - - -
H321C  .04(1)  .318(3)   .320(2)   .14000   Uiso  ? ?
H322A  .25(1)  .272(4)   .475(3)   .19000   Uiso  ? ?
H322B  .34976  .22118    .40954    .19000   Uiso
                                             calc C322
;
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   Example 2 - based on data set DPTD of Yamin,
    Suwandi, Fun, Sivakumar & bin Shawkataly
    [Acta Cryst. (1996), C52, 951-953].
;
;   loop_
    _atom_site_label
    _atom_site_chemical_conn_number
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
    _atom_site_U_iso_or_equiv
S1  1  0.74799(9)  -0.12482(11) 0.27574(9) 0.0742(3)
S2  2  1.08535(10)  0.16131(9)  0.34061(9) 0.0741(3)
N1  3  1.0650(2)   -0.1390(2)   0.2918(2)  0.0500(5)
C1  4  0.9619(3)   -0.0522(3)   0.3009(2)  0.0509(6)
# - - - - data truncated for brevity - - - -
;
    _definition
;    Data items in the ATOM_SITE category record
     details about the atom sites in a crystal
     structure, such as the positional
     coordinates, atomic displacement parameters,
     and magnetic moments and directions.
;
```

Fig. 2.5.5.5. DDL1 overview of a category of items.

2.5.5.9. Definition example 9: enumeration states

The last example, in Fig. 2.5.5.9, shows the definition of an item whose value is restricted to a predictable set of values known as enumeration states. The attributes `_enumeration` and `_enumeration_detail` are used to specify which enumeration states are permitted for the defined data item. Only one of these states may appear as the value for the defined item in a CIF. The attribute `_enumeration_default` specifies the state value that is used if an item is not instantiated.

```
data_atom_site_label
    _name                       '_atom_site_label'
    _category                     atom_site
    _type                         char
    _list                         yes
    _list_mandatory               yes
    loop_ _list_link_child
                  '_atom_site_aniso_label'
                  '_geom_angle_atom_site_label_1'
                  '_geom_angle_atom_site_label_2'
                  '_geom_angle_atom_site_label_3'
                  '_geom_bond_atom_site_label_1'
                  '_geom_bond_atom_site_label_2'

    loop_ _example    C12     Ca3g28     Fe3+17
                      H*251  boron2a  C_a_phe_83_a_0
                      Zn_Zn_301_A_0
    _definition
;    The _atom_site_label is a unique identifier for
     a particular site in the crystal.
;
```

Fig. 2.5.5.6. DDL1 definition of a 'mandatory' data item.

```
data_atom_site_aniso_label
    _name                     '_atom_site_aniso_label'
    _category                   atom_site
    _type                       char
    _list                       yes
    _list_link_parent         '_atom_site_label'
    _definition
;   Anisotropic atomic displacement parameters are
    usually looped in a separate list. If this is the
    case, this code must match the _atom_site_label
    of the associated atom in the atom coordinate
    list and conform with the same rules described
    in _atom_site_label.
;
```

Fig. 2.5.5.7. DDL1 definition of a 'parent' data item.

```
data_atom_site_aniso_U_
    loop_ _name                 '_atom_site_aniso_U_11'
                                '_atom_site_aniso_U_12'
                                '_atom_site_aniso_U_13'
                                '_atom_site_aniso_U_22'
                                '_atom_site_aniso_U_23'
                                '_atom_site_aniso_U_33'
    _category                   atom_site
    _type                       numb
    _type_conditions            su
    _list                       yes
    _list_reference           '_atom_site_aniso_label'
    _related_item             '_atom_site_aniso_B_'
    _related_function           conversion
    _units                     A^2^
    _units_detail             'angstroms squared'
    _definition
;   These are the standard anisotropic atomic
    displacement components in angstroms squared.
;
```

Fig. 2.5.5.8. DDL1 definition showing 'related' data items.

2.5.6. DDL1 attribute descriptions

This section provides an overview of the different attributes that make up the core data dictionary language DDL1. A more detailed description of attributes is given in the DDL1 dictionary in Chapter 4.9. In this dictionary the attributes are used to define themselves!

```
data_atom_site_adp_type
    _name                       '_atom_site_adp_type'
    _category                     atom_site
    _type                         char
    _list                         yes
    _list_reference             '_atom_site_label'
    loop_ _enumeration
          _enumeration_detail
                        Uani    'anisotropic Uij'
                        Uiso    'isotropic U'
                        Uovl    'overall U'
                        Umpe    'multipole expansion U'
                        Bani    'anisotropic Bij'
                        Biso    'isotropic B'
                        Bovl    'overall B'
    _definition
;   A standard code used to describe the type of
    atomic displacement parameters used for the site.
;
```

Fig. 2.5.5.9. DDL1 definition showing enumeration states.

DDL1 attributes are most easily understood when considered in groups with common descriptive functions. There are five basic functional groups of attributes, which perform the definition tasks of identifying, describing, typing, relating and registering data items.

2.5.6.1. Identification attributes

Establishing the identity of a data item is a primary function of a dictionary. In a data instantiation each item is recognized by a unique code, known as its data name or tag. This tag provides the most fundamental level of data validation.

The 'identification' attribute in DDL1 dictionaries is **_name**, and appears in a definition as

_name '<*dataname*>**'**

The tag of the defined item is the value of **_name** and, because it starts with an underscore, it must always be bounded by quotes to prevent it from being interpreted as the start of another tag–value pair. If there is more than one data item in a definition, as in the case of an irreducible set of items, the data names are entered as a list starting with the statement

loop_ _name

(see the example in Fig. 2.5.5.8).

The presence of the identification attribute **_name** in a definition is mandatory. Its value, the name of the defined items, provides for spelling validation. Note that if a data item in a CIF is not defined in a dictionary the CIF is still valid, although CIF parsers that employ dictionaries as part of the scanning process usually flag undefined items. The accepted practice to date is that undefined data are largely ignored by a dictionary validation process. In contrast, the need for certain defined items to be present in CIF data can be crucial because of list dependencies. The attributes that specify data relationships and links are considered in Section 2.5.6.4.

2.5.6.2. Descriptive attributes

Three DDL1 attributes are used to provide text descriptions of a defined data item. These are present in a dictionary for human readability, browser software or for the production of text dictionaries such as those in Part 4. This group of attributes is not machine interpretable.

_definition
_example
_example_detail

The **_definition** attribute provides a text description and as such is the primary semantic content in a defined data item. It may also be used to provide supplementary information about other machine-parsable attributes in the definition (see the definition in Fig. 2.5.5.3). The attributes **_example** and **_example_detail** are used to show typical instantiations of the defined item, as also shown in Fig. 2.5.5.6.

2.5.6.3. Typing attributes

This class of attributes is used to specify the fundamental characteristics of data items and how they may be instantiated. These attributes are machine-parsable and of particular importance in the validation of CIF data. They are

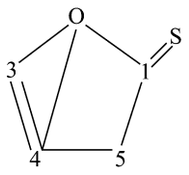
_enumeration
_enumeration_default
_enumeration_detail
_enumeration_range
_list
_list_level
_type
_type_conditions
_type_construct
_units
_units_detail

Enumeration attributes are used to specify restricted values, or 'states', of a data item (see the example in Fig. 2.5.5.9). They are not applicable in a definition of an item with unrestricted values. The attributes **_enumeration** and **_enumeration_detail** are used in definitions to specify permitted states and their descriptions. For instance, in a definition of atomic element symbols these attributes would be used to list the IUPAC 'element symbols' and 'element names'. When a data item is restricted to an ordinal set of values, the attribute **_enumeration_range** is used to define the minimum and maximum values of the logical sequence in the format <*min*>:<*max*> The attribute **_enumeration_default** may be used to specify the default value if an item is not present in a data instantiation.

List attributes specify how, and when, data items are used in a (looped) list. The attribute **_list** has a value of `yes` if a defined item must be in a list, and `no` if it must not. A value of `both` allows for both uses. The attribute **_list_level** specifies the nesting level of a list defined item. Only level 1 data are allowed in a CIF (see Chapter 2.2). Higher-nested list levels are permitted for MIF data (Allen *et al.*, 1995; see Chapter 2.4 for a description of the MIF format). Fig. 2.5.6.1 shows MIF data specifying a 2D chemical molecular diagram. The first four items in the category DISPLAY are in a level 1 list and the next two items in the category DISPLAY_CONN are in a level 2 list. Note that, according to the STAR File syntax, the nest level is automatically incremented after the first four values, and only decremented when a **stop_** signal is encountered.

Additional **_list** attributes for specifying relational dependencies between data items are described in the next attribute group.

Type attributes are used to specify the form of data. The attribute **_type** is restricted to the states `numb`, `char` and `null` that code for a number, a text string and a dictionary descriptor, respectively. The implications of this typing, in terms of how numbers or character strings are interpreted, is not specified by the CIF syntax. That is, a 'number' is simply a string of characters having one of several possible representations (*e.g.* as an integer, a floating point or in scientific notation) and it is up to the parsing software to interpret these strings appropriately. The same applies in the treatment of text strings defined as 'character'. Such strings may be bounded




```
loop_
    _display_id
    _display_symbol
    _display_coord_x
    _display_coord_y
    loop_
        _display_conn_id
        _display_conn_symbol
            1  1    180  110    2 1b    stop_
            2  O    140  140    3 1b    4 1b    stop_
            3  3    100  110    4 2b    stop_
            4  4    120   60    5 1b    stop_
            5  5    160   60    1 1b    stop_
            6  S    205  140    1 2b    stop_
```

(*a*)

```
data_display_coord_
    loop_ _name              '_display_coord_x'
                             '_display_coord_y'
    _type                     numb
    _list                     yes
    _list_level               1
    _definition
;   The projected coordinates of the display
    object identified in _display_object and
    _display_symbol.
;

data_display_conn_symbol
    _name                   '_display_conn_symbol'
    _type                     char
    _list                     yes
    _list_level               2
    _definition
;   The symbol code for the bond connection to the
    displayed objects identified by _display_id and
    _display_conn_id. The symbol codes and
    descriptions are stored in the validation file
    'mif_core_bonds.val'.
;
```

(*b*)

Fig. 2.5.6.1. (*a*) Hypothetical example and (*b*) the typical definition of nested items.

by white space, quotation characters or, in the case of multi-line text, by semicolon characters in column 1 of a line.

The attribute **_type_conditions** is used to identify application-specific conditions that apply to the typing of data items. This attribute is used to alert parsing software of string constructions that may disrupt the normal identification and validation of data types. The permitted attribute states are `none`, `esd` (and its preferred synonym `su`) and `seq`. In the definition example in Fig. 2.5.5.3 the code `esd` signals that cell-length values are measurements and may have a standard uncertainty value appended in parentheses. Fig 2.5.5.8 indicates the use of the code `su` for the same purpose. A full description of **_type_conditions** states is given in the DDL1 dictionary in Chapter 4.9.

If a data item consists of a concatenation of values, the attribute **_type_construct** may be used to specify the encoding algorithm for the component values. The language used for this algorithm is POSIX regex (IEEE, 1991). The specification of **_type_construct** can include the data names of other defined items. Validation software interprets this attribute as format specifications and constraints, and expands the embedded data items according to their

```
data_publ_date
    _name                   '_publ_date'
    _type                    char
    _type_construct
        (_publ_year)/(_publ_month)/(_publ_day)
    _definition
;   Specifies the syntax for declaring the
    publication date.
;

data_publ_year
    _name                   '_publ_year'
    _type                   char
    _type_construct         19|20[0-9][0-9]
    _definition
;   Specifies how the publication year will
    appear. Syntax is valid only for publication
    in the 20th and 21st centuries!
;
```

Fig. 2.5.6.2. Definition example: type constructions.

separate definition. For example, the chronological date is a composite of day, month and year. These may be represented in many different ways and the **_type_construct** attribute enables a specific date representation to be defined. The two definitions shown in Fig. 2.5.6.2 illustrate how a date value of the construction '1995/03/25' may be embedded into the dictionary definitions (the month and day definitions have been omitted here for brevity).

Units attributes specify the measurement units permitted for a numerical data item. The attribute **_units** specifies a code that uniquely identifies the measurement units. A description of the units identified by the code is given by the attribute **_units_detail**. Typical uses of this attribute are shown in Fig. 2.5.5.8.

2.5.6.4. Relational attributes

This class of attributes is used to describe special relationships and dependencies between data items. These attributes are machine parsable and are formally defined in the DDL1 dictionary in Chapter 4.9. They are

```
_category
_list_link_child
_list_link_parent
_list_mandatory
_list_reference
_list_uniqueness
_related_item
_related_function
```

The attribute **_category** is used to specify to which group, or basis set, a data item belongs. The value of this attribute is a name string that identifies the group and it is usually the leading part of the tags for all items in this group. Data items in a list must have the same category value. Data items in the same category may, however, be divided into different lists, provided each list contains an appropriate key data item (see **_list_reference** below).

List-link attributes are used to specify dependencies between data items in different lists. The attribute **_list_link_parent** identifies an item in another list from which the defined item was derived. The parent data item, or items in the case of irreducible sets, must have a unique value within its own list. The **_list_link_child** attribute is declared in the definition of a parent item and identifies items in other lists that depend implicitly on the defined data item being present in the same data instantiation. The functionalities of these two attributes mirror each other. The parent item must be present in any data instantiation containing the child items, but not the converse. The definition examples in Figs 2.5.5.6 and 2.5.5.7 illustrate the use of these attributes.

The attributes **_list_mandatory** and **_list_reference** are also connected to each other. The former signals (with values of yes or no) whether the presence of a data item is essential to preserving the validity of a list of items. The latter attribute identifies the data item in the list that provides the unique key value to each packet or row of items in a list. A packet is made up of listed values, one for each item in the name list (*i.e.* the packet size matches the number of data names at the head of the list). The **_list_reference** attribute identifies items that are the keys to specific packets in the list. In Example 2 of Fig. 2.5.5.5 the key item is **_atom_site_label** and the listed labels S1, S2, N1 and C1 must be unique.

The attribute **_list_uniqueness** is used to identify items that must be unique for a list to be valid and accessible. This attribute is similar to **_list_reference** except that it appears only in a definition in which **_list_mandatory** is set to yes. This simplifies validation because it may be used as the placeholder for all items that jointly identify the uniqueness of a list packet. This is in contrast to the attribute **_list_reference**, which appears in the definition of every item dependent on this item.

Relational attributes are used to link equivalent data. The **_related_item** attribute identifies items related to the defined data item. The nature of this relationship is specified with the **_related_function** attribute according to the restricted value states of alternate, convention, conversion and replace. The definition of these states is detailed in Chapter 4.9. Relational attributes are used to provide equivalent data items, to replace definitions when definitions are superseded or to change access pathways. These facilities are for archives, because they enable old data to be accessed and the associated definitions to remain in a dictionary even when superseded by new definitions. The old and new definitions are linked by these attributes so that all related data items can be validated and accessed.

2.5.6.5. Dictionary registration attributes

This class of attributes is used to register the dictionary version and audit information. They are

_dictionary_history
_dictionary_name
_dictionary_update
_dictionary_version

Dictionary attributes are used to record the creation and update history of a dictionary. The attribute **_dictionary_history** specifies the entry and update information of the dictionary, and **_dictionary_name** specifies the generic name of the electronic file containing the dictionary (the actual name of the file can vary from site to site). The attributes **_dictionary_version** and **_dictionary_update** specify the version number and the date of the last change in the dictionary. Both items represent important external reference information. An example application of these attributes is shown in Fig. 2.5.5.2.

References

Allen, F. H., Barnard, J. M., Cook, A. F. P. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Bourne, P. E., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. & Fitzgerald, P. M. D. (1997). *Macromolecular Crystallographic Information File. Methods Enzymol.* **277**, 571–590.

Gray, P. M. D., Kulkarni, K. G. & Paton, N. W. (1992). *Object oriented databases*. New York: Prentice Hall.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

IEEE (1991). *IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and utilities*, Vol. 1, IEEE Standard 1003.2-1992. New York: The Institute of Electrical and Electronic Engineers, Inc.

Kim, W. (1990). *Introduction to object oriented databases*. Boston: MIT Press.

Westbrook, J. D. & Hall, S. R. (1995). *A dictionary description language for macromolecular structure.* http://ndbserver.rutgers.edu/mmcif/ddl/.

2.6. Specification of a relational dictionary definition language (DDL2)

BY J. D. WESTBROOK, H. M. BERMAN AND S. R. HALL

2.6.1. Introduction

The dictionary definition language version 2 (DDL2) presented here extends the DDL1 version (Hall & Cook, 1995) currently used by the IUCr for the description of data items common to all crystallographic studies (*i.e.* core items). The DDL2 extensions were introduced primarily to address two issues arising during the development of a CIF dictionary for the terminology of macromolecular crystallography: the need to accurately describe the hierarchical nature of macromolecular structure and associated structural features, and the desire to encode dictionary definitions in a manner that would permit more detailed software-driven validation.

The decision not to use DDL1 for the macromolecular CIF dictionary (mmCIF) was not made lightly. The Working Group responsible for the development of the mmCIF dictionary spent three years building a dictionary version within a DDL1 framework. When presented at the first mmCIF workshop in York in 1993 (Chapter 1.1), it was criticized as lacking the rigour to be usefully interpreted by software. In particular, the draft dictionary lacked machine-interpretable relationships between the components of macromolecular structure, the components of structure and structural properties, and the components of structure and the experimental description. These relationships are important to a complete description of macromolecular structural data, and need to be present in a dictionary in a form that permits software to navigate and validate them.

Following the York workshop, the Working Group set about redesigning the framework of the data model used to organize the dictionary definitions. Initially there was significant interest in adopting a more object-oriented data model so as to match closely the object-oriented characteristics of macromolecular structure. However, an object-oriented model would depart significantly from the organization of the core CIF dictionary based on DDL1, and interoperability between the two approaches would be likely to be problematic.

The new DDL concepts were presented at the Brussels mmCIF workshop in 1994. This approach, which later became known as DDL2 (Westbrook & Hall, 1995), employs a largely relational data model that adhered more closely to the data model implicit in the core DDL1. DDL2 added new elements to expand the DDL1 concept of a data category. Categories in DDL2 are fully realized definitional elements that have their own set of attributes (*e.g.* definitions and examples). DDL2 also added data elements to explicitly define parent–child relationships between data items within a hierarchy of categories. It was demonstrated that using this conservative relational data model it was possible to accurately describe the content in the mmCIF dictionary.

Affiliations: JOHN D. WESTBROOK and HELEN M. BERMAN, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia.

Example 2.6.2.1. *Typical mmCIF definition save frame.*

```
save__exptl.details
  _item_description.description
; Any special information about the experimental work
  prior to the intensity measurement. See also
  _exptl_crystal.preparation.
;
  _item.name                  '_exptl.details'
  _item.category_id           exptl
  _item.mandatory_code        no
  _item_aliases.alias_name    '_exptl_special_details'
  _item_aliases.dictionary    cif_core.dic
  _item_aliases.version       2.0.1
  _item_type.code             text
save_
```

In view of the importance of maintaining continuity between small- and large-molecule crystallography, the extensions in DDL2 have been introduced in a manner that provides the greatest degree of backward compatibility with applications and dictionaries developed on the core DDL1. Like DDL1, DDL2 dictionaries and data files are fully compliant with the underlying syntax rules of the Self-defining Text Archive and Retrieval (STAR) File (Hall, 1991; Hall & Spadaccini, 1994). Although DDL2 uses a different convention to name data items, an alias feature in DDL2 is used to maintain a correspondence with the published data-item names in the core CIF dictionary (Hall *et al.*, 1991; Chapter 4.1).

Since its introduction in 1994, DDL2 has been used to develop the mmCIF dictionary and this has been adopted as the data exchange standard of the Protein Data Bank, the international repository of three-dimensional macromolecular structure data (see Chapter 5.5).

2.6.2. The DDL2 presentation

The syntax used in mmCIF data files and DDL2 dictionaries is a subset of the STAR syntax. Definitions in DDL2 dictionaries are encapsulated in named save frames (see Chapter 2.1 for a description of the STAR File syntax). A save frame is a syntactical element that begins with the `save_` directive and is terminated by another `save_` directive. Save frames are named by appending a text string to the `save_` directive. In the mmCIF dictionary, save frames are used to encapsulate item and category definitions. The mmCIF dictionary is composed of a data block containing thousands of save frames, where each save frame contains a different definition. Save frames appear in DDL2 dictionaries but they are not used in data files. Save frames may not be nested.

DDL2 dictionary definitions typically contain a small number of items that specify the essential features of the item. Example 2.6.2.1 shows a save frame containing the mmCIF definition of the data item `_exptl.details`.

The example definition includes: a description or text definition, the name and category of the item, a code indicating that the item is optional (not mandatory), the name of a related definition in the core CIF dictionary, and a code specifying that the data type is text. A more detailed description of the elements of the dictionary definitions is presented in the following sections.

2.6.3. Overview of the elements of DDL2

The elements of DDL2 provide the organizational framework for building data dictionaries like mmCIF. The role of the DDL is to define which data items may be used to construct the definitions in the data dictionary, and also to define the relationships between these defining data items. The DDL2 attributes are defined in the dictionary presented in Chapter 4.10.

A dictionary language contains no specific information about a discipline, such as macromolecular crystallography; rather, it defines the data items that can be used to describe a discipline. The contents of the mmCIF dictionary are metadata, or data about data. The contents of the DDL are meta-metadata, the data defining the metadata. By design DDL2, like its predecessor DDL1, is quite generic. It defines data items that describe the general features of a data item like a textual description, a data type, a set of examples, a range of permissible values, or perhaps a discrete set of permitted values. Consequently, data modelling using a DDL can be applied in many fields.

The lowest level of organization provided by DDL2 is the description of an individual data item. Collections of related data items are organized in categories. Categories are essentially tables in which each repetition of the group of related items adds a row. The terms category and data item are used here in order to conform with the previous use of these terms by STAR and CIF applications; these terms could be replaced by relation and attribute (or table and column) commonly used to describe the relational model which underlies DDL2.

Within a category, the set of data items determining the uniqueness of their group are designated as key items in the category. No data-item group in a category is allowed to have a set of duplicate values of its key items. Each data item is assigned membership in one or more categories. Parent–child relationships may be specified for items belonging to multiple categories. A parent–child relationship identifies cases in which the same data item, often an important identifier, occurs in different categories. These relationships permit the expression of the very complicated hierarchical data structures required to describe macromolecular structure.

Other levels of organization in addition to category are also supported. Related categories may be collected together in category groups and parent relationships may be specified for these groups. This higher level of association provides a vehicle for organizing a large complicated collection of categories into smaller more relevant and potentially interrelated groups. This effectively provides a chaptering mechanism for large and complicated dictionaries, like mmCIF. Within the level of a category, subcategories of data items may be defined among groups of related data items. The subcategory provides a mechanism for identifying, for example, that the data items month, day, and year collectively define a date.

For categories, subcategories and items, methods may be specified. Methods are computational procedures that are defined and expressed in a programming language (*e.g.* C/C++, Perl or Java) and stored within a dictionary. Among other things, these dictionary methods may be used to calculate a missing value or to check the validity of a particular value.

The highest levels of data organization provided by DDL2 are the data block and the dictionary. The dictionary level collects a set of related definitions into a single unit and provides the attributes for a detailed revision history of the collection.

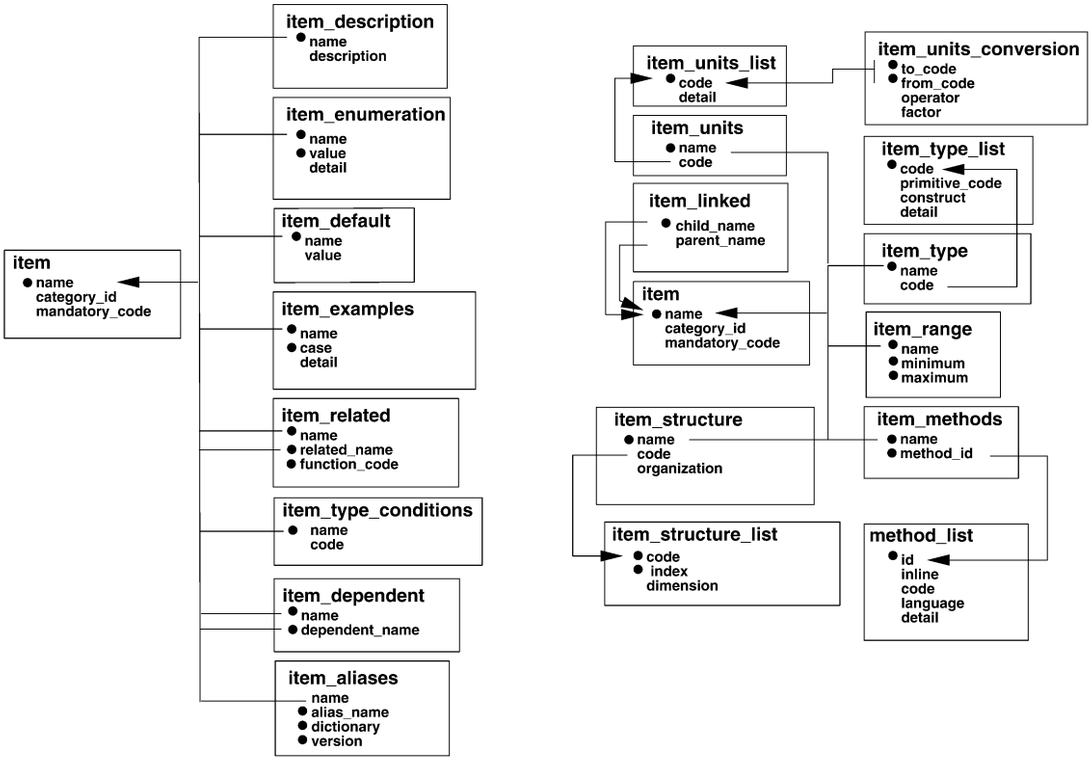


Fig. 2.6.4.1. DDL2 attributes used to specify item information. In this figure, boxes surround data items belonging to the same category. Category identifiers are given in a large typesize and item names are given in a smaller typesize. Parent–child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are marked with a bullet.

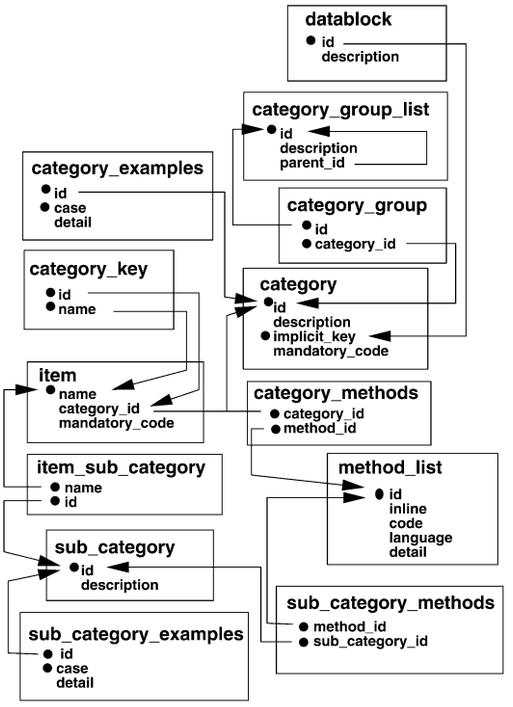


Fig. 2.6.4.2. DDL2 attributes used to specify category, subcategory and category group information. Category identifiers are given in a large typesize and item names are given in a smaller typesize. Parent–child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are marked with a bullet.

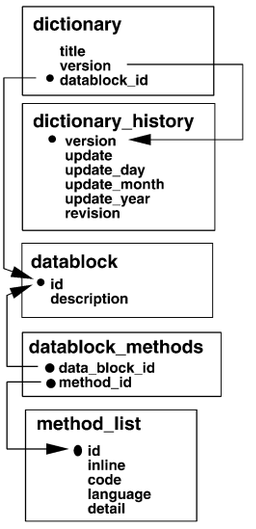


Fig. 2.6.4.3. DDL2 attributes used to specify dictionary and data-block information. Category identifiers are given in a large typesize and item names are given in a smaller typesize. Parent–child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are marked with a bullet.

Fig. 2.6.4.2 illustrates the organization of attributes describing categories, subcategories and category groups. Similarly, Fig. 2.6.4.3 shows the organization of DDL2 attributes at the datablock and dictionary level. All of these attributes are discussed in terms of their application in building data dictionaries in the next section.

2.6.4. DDL2 organization

Figs 2.6.4.1–2.6.4.3 provide schematic illustrations of the definitional features provided by DDL2. These figures represent the elements of the DDL in terms of its own language constructs (*i.e.* categories and the relationships between attributes within those categories). This self-defining presentation has the important consequence of validating the internal consistency of the DDL data model.

Fig. 2.6.4.1 shows the organization of the attributes available to define each data item. These include: a description, examples, data type, allowed values and ranges, default values, internal structural features (*e.g.* vector and matrix properties), units, and other dependency relationships. These DDL attributes are shown as a collection of DDL categories enclosed in boxes in the figure. For instance, the description or textual definition for a data item is specified in a category named ITEM_DESCRIPTION. This DDL category contains the attributes 'name' and 'description'. The attribute 'name' corresponds to the DDL data item **`_item_description.name`**. This item is the key item in the category named ITEM_DESCRIPTION. In Fig. 2.6.4.1 this is denoted by a bullet. The name attribute in the ITEM_DESCRIPTION category is related to the parent definition of this data item in the category named ITEM. This is reflected in Fig. 2.6.4.1 by the line pointing to the parent data item.

The data-block level ties the contents of a dictionary to the **`data_`** section in which it is contained. The identifier for the data block and hence the dictionary is added implicitly to the key of each category. This builds into the data model a convenient means for distinguishing similar information recorded in separate data blocks. This feature is important in organizing the results from different crystallographic experiments, each being reported as a separate block of data.

2.6.5. DDL2 dictionary applications

In this section, several examples are presented which illustrate how the elements of the DDL are used to build dictionary definitions. Example 2.6.5.1 shows the definition of the **`_citation.journal_abbrev`** data item from the mmCIF dictionary.

The category ITEM_DESCRIPTION holds a text description of each data item. The category ITEM holds the item name, category name and a code indicating whether this item is mandatory in any row of this category. The value of the mandatory code is either `yes`, `no` or `implicit`. The `implicit` value is used to indicate that a value is required for the item but it can be derived from the context of the definition and need not be specified. This feature is most often used in DDL2 dictionaries to avoid re-specifying data-item names in each category since these values can be derived from the name of the save frame enclosing the definition. The value of the **`_item.name`** in the above example is enclosed in quotation marks. This is a requirement of the STAR syntax so that a value containing a data name is not mistaken for a dictionary attribute.

The mmCIF dictionary contains a superset of the definitions that were originally defined in the core CIF dictionary. In order to maintain backward compatibility with original definitions, the ITEM_ALIASES category was introduced to hold the item name, dictionary name and version in which the original definition of an item was published. In this example, the data name used in the core dictionary differs from the example definition only in the period that distinguishes the category and attribute portions of the item name.

The category ITEM_TYPE holds a reference to a data type specified in the ITEM_TYPE_LIST category. A reference to the data type is used here rather than a detailed data-type description in order to avoid repeating the description for other data items. A single list

Example 2.6.5.1. *A rather simple data definition.*

```
save_citation.journal_abbrev
   _item_description.description
; Abbreviated name of the cited journal given in
   the Chemical Abstracts Service Source Index.
;
   _item.name                '_citation.journal_abbrev'
   _item.category_id           citation
   _item.mandatory_code        no
   _item_aliases.alias_name
                             '_citation_journal_abbrev'
   _item_aliases.dictionary  cif_core.dic
   _item_aliases.version     2.0.1
   _item_type.code           line
   _item_examples.case       'J. Mol. Biol.'
save_
```

of data types and associated regular expressions is stored in the ITEM_TYPE_LIST category and this may be referenced by all of the definitions in the dictionary. In the mmCIF dictionary, the codes that are used to describe the data types are generally easy to interpret. In this example, the type code 'line' indicates that a single line of text will be accepted for this data item.

Descriptive examples of data items can be included in the ITEM_EXAMPLES category. In Example 2.6.5.1, one value, `'J. Mol. Biol.'`, is specified, but multiple examples can be provided using a **`loop_`** directive.

Other DDL item attributes are illustrated in the mmCIF definitions for the items **`_cell.length_a`** and **`_cell.length_a_esd`** in Example 2.6.5.2. Some data items are only meaningful as part of a complete set. The ITEM_DEPENDENT category is used to store this type of information. Those additional data items within the irreducible set are listed in this category. In Example 2.6.5.2, the cell lengths in the *b* and *c* directions are defined as dependent items of the cell length in the *a* direction.

The permissible ranges of values for a numerical data item are stored in the ITEM_RANGE category. Each boundary condition is defined as the non-inclusive range between a pair of minimum and maximum values. If multiple boundary conditions are specified using the **`loop_`** directive, then each condition must be satisfied. A discrete boundary value may be set by assigning the desired boundary value as both the maximum and minimum value. In the above example, the permissible cell-length range is defined as greater than or equal to zero, where the latter boundary condition is specified by setting both extrema as zero.

A number of special relationships may be defined between data items. For some relationships which occur frequently, the source or function of the relationship has been standardized. In the example above, this feature is used to identify that the **`_cell.length_a_esd`** is the standard uncertainty (estimated standard deviation) of **`_cell.length_a`**. The recognized relationships are fully described in the DDL definition of the data item **`_item_related.function_code`** in category ITEM_RELATED. The current list includes the kinds of relationships in Table 2.6.5.1.

Sets of data items within a category may be collected into named subcategories. ITEM_SUB_CATEGORY is used to store the subcategory membership of a data item. In the above example, item **`_cell.length_a`** is added to the subcategory CELL_LENGTH. The items **`_cell.length_b`** and **`_cell.length_c`** are similarly added to this subcategory in their definitions.

The ITEM_UNITS category holds the name of the system of units in which an item is expressed. The name assigned to **`_item_units.code`** refers to a single list of all of the unit types used in the dictionary. This list is stored in the category ITEM_UNITS_LIST. Conversion factors between different systems of units are provided in the data table stored in the ITEM_UNITS_CONVERSION category.

Example 2.6.5.2. *Definition of a data item that has dependencies and associated items.*

```
save_cell.length_a
   _item_description.description
; Unit-cell length a corresponding to the structure
   reported in angstroms.
;
   _item.name                    '_cell.length_a'
   _item.category_id               cell
   _item.mandatory_code            no
   _item_aliases.alias_name      '_cell_length_a'
   _item_aliases.dictionary        cif_core.dic
   _item_aliases.version           2.0.1
    loop_
   _item_dependent.dependent_name
                                 '_cell.length_b'
                                 '_cell.length_c'
    loop_
   _item_range.maximum
   _item_range.minimum             .    0.0
                                 0.0    0.0
   _item_related.related_name    '_cell.length_a_esd'
   _item_related.function_code     associated_esd
   _item_sub_category.id           cell_length
   _item_type.code                 float
   _item_type_conditions.code      esd
   _item_units.code                angstroms
save_

save_cell.length_a_esd
   _item_description.description
; The standard uncertainty (estimated standard
   deviation) of _cell.length_a.
;
   _item.name                    '_cell.length_a_esd'
   _item.category_id               cell
   _item.mandatory_code            no
   _item_default.value             0.0
    loop_
   _item_dependent.dependent_name
                                 '_cell.length_b_esd'
                                 '_cell.length_c_esd'
   _item_related.related_name    '_cell.length_a'
   _item_related.function_code     associated_value
   _item_sub_category.id           cell_length_esd
   _item_type.code                 float
   _item_units.code                angstroms
save_
```

Example 2.6.5.3 shows the definition of the CELL category from the mmCIF dictionary. The name and textual description of a category are stored in the category named CATEGORY. The item named **`_category.mandatory_code`** indicates whether the category must appear in any data block based on this dictionary.

The list of data items that uniquely identify each row of a category are stored in the CATEGORY_KEY category. In the example above, the item **`_cell.entry_id`** is defined as the category key. This item is a reference to the top-level identifier in the mmCIF dictionary, **`_entry.id`**. Because only a single entry may exist within an mmCIF data block, this choice of category key specifies that only a single row may exist in the CELL category.

Membership in category groups is stored in the category named CATEGORY_GROUP. Each category group must have a corresponding definition in the category CATEGORY_GROUP_LIST. In the above example, the CELL category is assigned to category groups **`cell_group`** and **`inclusive_group`**. The former contains categories that describe properties of the crystallographic cell and the latter includes all the categories in the mmCIF dictionary. Organizing categories in category groups is a convenient means of providing a high-level organizational structure for a complex dictionary.

Complete and annotated examples of a category are stored in the CATEGORY_EXAMPLES category. The text of the category example is stored in the item **`_category_examples.case`** and any associated annotation is stored in the item **`_category_examples.detail`**.

Table 2.6.5.1. *Relationships defined by* **_item_related.function_code**

Code	Meaning
alternate	The item identified in **_item_related.related_name** is an alternative expression in terms of its application and attributes to the item in this definition
alternate_exclusive	The item identified in **_item_related.related_name** is an alternative expression in terms of its application and attributes to the item in this definition; only one of the alternative forms may be specified
convention	The item identified in **_item_related.related_name** differs from the defined item only in terms of a convention in its expression
conversion_constant	The item identified in **_item_related.related_name** differs from the defined item only by a known constant
conversion_arbitrary	The item identified in **_item_related.related_name** differs from the defined item only by an arbitrary constant
replaces	The defined item replaces the item identified in **_item_related.related_name**
replacedby	The defined item is replaced by the item identified in **_item_related.related_name**
associated_value	The item identified in **_item_related.related_name** is meaningful when associated with the defined item
associated_esd	The item identified in **_item_related.related_name** is the standard uncertainty (estimated standard deviation) of the defined item

Example 2.6.5.3. *Definition of an mmCIF category.*

```
save_CELL
   _category.description
; Data items in the CELL category record details
  about the crystallographic cell parameters.
;
  _category.id                    cell
  _category.mandatory_code        no
  _category_key.name            '_cell.entry_id'
   loop_
  _category_group.id             'inclusive_group'
                                 'cell_group'
  _category_examples.detail
# - - - - - - - - - - - - - - - - - - - - - - - - -
;
Example 1 - based on PDB entry 5HVP and laboratory
            records for the structure corresponding
            to PDB entry 5HVP
;
   _category_examples.case
;
_cell.entry_id                            '5HVP'
_cell.length_a                            58.39
_cell.length_a_esd                         0.05
_cell.length_b                            86.70
_cell.length_b_esd                         0.12
_cell.length_c                            46.27
_cell.length_c_esd                         0.06
_cell.angle_alpha                         90.00
_cell.angle_beta                          90.00
_cell.angle_gamma                         90.00
_cell.volume                             234237
_cell.details
    ; The cell parameters were refined every twenty
      frames during data integration. The cell
      lengths given are the mean of 55 such
      refinements;  the esds given are the root mean
      square deviations of these 55 observations
      from that mean.
    ;
;
save_
```

Example 2.6.5.4 illustrates the definition of a pair of mmCIF categories, CITATION and CITATION_AUTHOR, which share a common data item, **_citation.id**. This example illustrates how an item that occurs in multiple categories may be defined. In the case of the citation identifier, **_citation.id**, the ITEM category is preceded by a **loop_** directive and within this loop all of the definitions of the citation identifier are listed. For instance, the citation identifier is also an item in category CITATION_AUTHOR, where it has the item name **_citation_author.citation_id**. For conformity with the manner in which the core CIF dictionary has been organized, a skeleton definition of the child data item **_citation_author.citation_id** has been included in the dictionary. In fact, this skeleton definition is formally unnecessary.

As a matter of style, the mmCIF dictionary generally defines all of the instances of a data item within the parent definition. Items that are related to the parent definition are also listed in the ITEM_LINKED category. The repetition of a data item in multiple categories gives rise to parent–child relationships between such definitions. These relationships are stored in the ITEM_LINKED category. In Example 2.6.5.4, this category stores the list of data items that are children of the citation identifier **_citation.id**. These include **_citation_author.citation_id**, **_citation_editor.citation_id** and **_software.citation_id**.

2.6.6. Detailed DDL2 specifications

DDL2 is presented here (Chapter 4.10) in the form of a dictionary that is defined in terms of its own definitional elements. This self-consistent description not only provides a prototype for other application dictionaries, but also provides a mechanism by which the consistency and relational integrity of the DDL data model can be independently verified. DDL2 defines a relatively simple set of organizational elements including data blocks, categories, category groups, subcategories and items. Data dictionaries (*e.g.* mmCIF) apply these elements provided by the DDL to describe the knowledge base of an application domain. The following sections provide detailed specifications of each definitional element of DDL2.

2.6.6.1. DDL2 definitions describing data items

In this section, the DDL2 categories that describe the properties of data items are presented. Figs 2.6.4.1 and 2.6.4.2 illustrate the organization of definitional elements in these categories.

2.6.6.1.1. *ITEM*

The category named ITEM is used to assign membership of data items to categories. This category forms the bridge between the category and data-item levels of abstraction. The key data item in this category is the full data-item name, **_item.name**. This name contains both the category and data-item identifiers, and is thus a unique identifier for the data item. The category identifier, **_item.category_id**, is included in this category as a separate mandatory data item. This has been done to provide an explicit reference to those categories that use the category identifier as a unique identifier.

One could alternatively use the category and item identifiers as the basis for this category rather than the concatenated form of the item name, and thus eliminate the redundant specification of the

Example 2.6.5.4. *Related categories linked by parent–child relationships.*

```
save_CITATION
_category.description
; Data items in the CITATION category record details
  about the literature cited as being relevant to
  the contents of the data block.
;
_category.id                  citation
_category.mandatory_code      no
_category_key.name          '_citation.id'
 loop_ _category_group.id    'inclusive_group'
                             'citation_group'
#      ---------  abbreviated definition  ----------
save_

save__citation.id
  _item_description.description
; The value of _citation.id must uniquely identify a
  record in the CITATION list. The _citation.id
  'primary' should be used to indicate the citation
  that the author(s) consider to be the most
  pertinent to the contents of the data block.
;
 loop_       _item.name
             _item.category_id
             _item.mandatory_code
'_citation.id'                   citation          yes
'_citation_author.citation_id'   citation_author   yes
'_citation_editor.citation_id'   citation_editor   yes
'_software.citation_id'          software          yes
_item_aliases.alias_name    '_citation_id'
_item_aliases.dictionary     cif_core.dic
_item_aliases.version        2.0.1
 loop_       _item_linked.child_name
             _item_linked.parent_name
'_citation_author.citation_id'   '_citation.id'
'_citation_editor.citation_id'   '_citation.id'
'_software.citation_id'          '_citation.id'
_item_type.code              code
 loop_ _item_examples.case       'primary'   '1'   '2'
save_

save_CITATION_AUTHOR
_category.description
; Data items in the CITATION_AUTHOR category record
  details about the authors associated with the
  citations in the CITATION list.
;
_category.id                  citation_author
_category.mandatory_code      no
 loop_
_category_key.name    '_citation_author.citation_id'
                      '_citation_author.name'
 loop_ _category_group.id     'inclusive_group'
                              'citation_group'
#      --------- abbreviated definition  ----------
save_

save__citation_author.citation_id
_item_description.description
; This data item is a pointer to _citation.id in the
  CITATION category.
;
_item.name              '_citation_author.citation_id'
_item.mandatory_code          yes
_item_aliases.alias_name
                       '_citation_author_citation_id'
_item_aliases.dictionary      cif_core.dic
_item_aliases.version         2.0.1
save_
```

category identifier. The full name has been used here in order to provide compatibility with existing applications.

The item category also includes a code to indicate whether a data item is mandatory in a category and therefore must be included in any tuple of items in the category. This code, **_item.mandatory_code**, may have three values: `yes`, `no` and `implicit`. This last named value indicates that the item is mandatory, but that the value of this item may be derived from the context. In the case of an item name or a category identifier, these values can be obtained from the current save-frame name. Implicit specification dramatically simplifies the appearance of each dictionary definition because it avoids the repeated declaration of item names and category identifiers that are basis components or the unique identifiers for most categories.

Although the data item **_item.name** is the basis for all of the item-level categories, its definition and properties need only be specified at a single point. Here, the data items that occur in multiple categories are defined only in the parent category. In certain situations, a child data item may be used in a manner which requires a description distinct from the parent data item. For instance, **_item_linked.parent_name** and **_item_linked.child_name** are both data-item names as well as children of **_item.name**, but clearly the manner in which these items are used in the ITEM_LINKED category requires additional description. It is important to note that although the design of this DDL supports the definition of data items in multiple categories within the parent category, it is also possible to provide separate complete definitions within each category.

2.6.6.1.2. *ITEM_ALIASES*

The DDL category ITEM_ALIASES defines the alias names that can be substituted for a data-item name. The alias mechanism also provides a means of identifying items by names other than those that follow the naming conventions used in this DDL. This feature should be used primarily to guarantee the stability of names defined in previously published dictionaries. The items **_item_aliases.name**, **_item_aliases.dictionary** and **_item_aliases.version** form the key for this category. The items **_item_aliases.dictionary** and **_item_aliases.version** are provided to distinguish between dictionaries and different versions of the same dictionary. Any number of unique alias names can be defined for a data item.

2.6.6.1.3. *ITEM_DEFAULT*

The DDL category ITEM_DEFAULT holds default values assigned to data items. Default data values are specified in item **_item_default.value**. Default values are assigned to data items that are not declared within a category. The key item for this category, **_item_default.name**, is a child of **_item.name**. A single default value may be specified for a data item.

2.6.6.1.4. *ITEM_DEPENDENT*

The ITEM_DEPENDENT category defines dependency relationships among data items within a category. Each data item on which a particular data item depends is specified as an item **_item_dependent.dependent_name**. For a data item to be considered completely defined, each of its dependent data items must also be specified.

2.6.6.1.5. *ITEM_DESCRIPTION*

The DDL category ITEM_DESCRIPTION holds a description for each data item. The key item for this category is **_item_description.name**, which is defined in the parent category ITEM. The text of the item description is held by data item **_item_description.description**. A single description may be provided for each data item.

2.6.6.1.6. *ITEM_ENUMERATION*

The DDL category ITEM_ENUMERATION holds lists of permissible values for a data item. Each enumerated value is specified in item **_item_enumeration.value**, each of which may have an associated description item **_item_enumeration.detail**. The combination of items **_item_enumeration.name** and **_item_enumeration.value** form the key for this category. The parent definition of the former item is defined in the category ITEM. Multiple unique enumeration values may be specified for each data item.

2.6.6.1.7. *ITEM_EXAMPLES*

The DDL category ITEM_EXAMPLES is provided to hold examples associated with individual data items. An example specification consists of the text of the example, **_item_examples.case**, and an optional comment item, **_item_examples.detail**, which can be used to qualify the example. Multiple examples may be provided for each item.

2.6.6.1.8. *ITEM_LINKED*

The ITEM_LINKED category defines parent–child relationships between data items. This provides the mechanism for specifying the relationships between data items that may exist in multiple categories. Link relationships are most commonly defined between key items, which form the keys for many different categories.

In the DDL definition, all child relationships are expressed within the parent category.

Because the item **_item_linked.parent_name** has been defined as an implicit item, the child relationships can be specified most economically in the parent category where the parent item name can be automatically inferred. If link relationships are specified in a child category, then both parent and child item names must be specified.

Both parent and child item names in this category are children of **_item.name**, which ensures that all link relationships can be properly resolved. However, it is possible to define cyclical link relationships within this category. Any implementation of this DDL category should include a method to check for the existence of such pathological cases.

2.6.6.1.9. *ITEM_METHODS*

The ITEM_METHODS category is used to associate method identifiers with data items. Any number of unique method identifiers may be associated with a data item. The method identifiers reference the full method definitions in the parent METHOD_LIST category.

2.6.6.1.10. *ITEM_RANGE*

The ITEM_RANGE category defines a restricted range of permissible values for a data item. The restrictions are specified as one or more sets of the items **_item_range.minimum** and **_item_range.maximum**. These items give the lower and upper bounds for a permissible range. To specify that an item value may be equal to the upper or lower bound or a range, the minimum and maximum values of the range are equated. The special STAR value indicating that a data value is not appropriate (denoted by a period, '.') can be used to avoid expressing an upper or lower bound value. When limits are applied to character data, comparisons are made following the collating sequence of the character set. When limits are applied to abstract data types, methods must be provided to define any comparison operations that must be performed to check the boundary conditions.

2.6.6.1.11. *ITEM_RELATED*

The ITEM_RELATED category describes specific relationships that exist between data items. These relationships are distinct from the parent–child relationships that are expressed in the category. The related item is identified as the item **_item_related.related_name** that is a child of **_item.name**.

Item relationships defined by **_item_related.function_code** in this category include some of the following (Table 2.6.5.1): an item is related to another item by a conversion factor; an item is a replacement for another item; an item is replaced by another item; an item is an alternative expression of an item; items which differ only in some convention of their expression; and items which express a set of related characteristics. One can also identify whether the declaration of an item is mutually exclusive with its alternative item. Multiple related items can be associated with each data item and multiple relationship codes can be specified for each related item.

2.6.6.1.12. *ITEM_STRUCTURE*

The ITEM_STRUCTURE category holds a code which identifies a structure definition that is associated with a data item. A structure in this context is a reusable matrix or vector definition declared in category ITEM_STRUCTURE_LIST. The data item **_item_structure.code** is a child of the item **_item_structure_list.code**. The item **_item_structure.code** provides an indirect reference into the list of structure-type definitions in category ITEM_STRUCTURE_LIST. The **_item_structure.organization** item describes the row/column precedence of the matrix organization.

2.6.6.1.13. *ITEM_STRUCTURE_LIST*

The ITEM_STRUCTURE_LIST category holds definitions of matrices and vectors that can be associated with data items. A component of the key for this category is **_item_type_list.code**, which is referenced by **_item_structure.code** to assign a structure type to a data item. The definition of a structure involves the specification of a length for each dimension of the matrix structure. The combination of items **_item_structure_list.code** and **_item_structure_list.index** forms the key for this category. The latter index item is the identifier for the dimension, hence multiple unique dimensions can be specified for each structure code. The length of each dimension is assigned to **_item_structure_list.dimension**.

2.6.6.1.14. *ITEM_SUB_CATEGORY*

The ITEM_SUB_CATEGORY category is used to assign subcategory membership for data items. A data item may belong to any number of subcategories. Each subcategory must be defined in a category named SUB_CATEGORY.

2.6.6.1.15. *ITEM_TYPE*

The ITEM_TYPE category holds a code that identifies the data type of each data item. The data item **_item_type.code** is a child of the item **_item_type_list.code**. Data-type definitions are actually made in the ITEM_TYPE_LIST parent category. The item **_item_type.code** provides an indirect reference into the list of data-type definitions in category ITEM_TYPE_LIST. This indirect

reference is provided as a convenience to avoid the redeclaration of the full data-type specification for each data item. The key item for this category is **_item_type.name**, which is defined in the parent category ITEM. Only one data type may be specified for a data item.

2.6.6.1.16. *ITEM_TYPE_CONDITIONS*

The category ITEM_TYPE_CONDITIONS defines special conditions applied to a data-item type. This category has been included in order to comply with previous applications of STAR and CIF. Since the constructions that are embodied in this category are antithetical to the data model that underlies DDL2, it is recommended that this category only be used for the purpose of parsing existing data files and dictionaries.

2.6.6.1.17. *ITEM_TYPE_LIST*

The ITEM_TYPE_LIST category holds the list of item data-type definitions. The key item in this category is **_item_type_list.code**. Data types are associated with data items by references to this key from the ITEM_TYPE category. One of the data-type codes defined in this category must be assigned to each data item.

The definition of a data type consists of the specification of the item's primitive type and a regular expression that defines the pattern that must be matched by any occurrence of the item. The primitive type code, **_item_type_list.primitive_code**, can assume values of `char`, `uchar`, `numb` and `null`. This code is provided for backward compatibility with STAR and CIF applications that employ loose data typing. The data item **_item_type_list.construct** holds the regular expression that must be matched by the data type. Simple regular expressions can be used to define character fields of restricted width, floating-point and integer formats.

Molecular Information File (MIF) applications (Allen *et al.*, 1995) have extended the notion of the regular expression to include data-item components. This permits the construction of complex data items from one or more component data items using regular expression algebra. These extended regular expressions are defined in the category ITEM_TYPE_CONDITIONS.

Example 2.6.6.1 illustrates the data types that are defined within this DDL. The DDL uses a number of character data types which have subtly different definitions. For instance, the data type identified as `code` defines a single-word character string; `char` extends the code type with the addition of a white-space character; and `text` extends the `char` type with the addition of a newline character. Two special character data types `name` and `idname` are used to define the full STAR data name and the STAR name components, respectively. The data type `any` is used to match any potential data type. This type is used for data items that may hold a variety of data types. The data type `int` is defined as one or more decimal digits and the `yyyy-mm-dd` type defines a date string.

2.6.6.1.18. *ITEM_UNITS*

The ITEM_UNITS category holds a code that identifies the system of units in which a data item is expressed. The data item **_item_units.code** is a child of the item **_item_units_list.code**. Unit definitions are actually made in the ITEM_UNITS_LIST parent category. The item **_item_units.code** provides an indirect reference into the list of data-type definitions in category ITEM_UNITS_LIST. This indirect reference is provided as a convenience to avoid the redeclaration of the full data-type specification for each data item. The key item for this category is **_item_units.name**, which is defined in the parent category ITEM. Only one type of unit may be specified for a data item.

Example 2.6.6.1. *The description of permitted data types in the DDL2 dictionary.*

```
#               DATA TYPE CONVERSION TABLE
#               --------------------------
#
      loop_
      _item_type_list.code
      _item_type_list.primitive_code
      _item_type_list.detail
      _item_type_list.construct


code  char  'A single word'               '[^\t\n "]*'
char  char  'A single line of text'       '[^\n]*'
text  char  'Text which may span lines'   '.*'
int   numb  'Unsigned integer data'       '[0-9]+'
name  uchar 'A data item name'
                '_[_A-Za-z0-9]+[.][][_A-Za-z0-9%/-]+'

idname uchar
     'A data item name component or identifier'
                                          '[_A-Za-z0-9]+'
any   char  'Any data type'                 '.*'
yyyy-mm-dd
      char  'A date format'
      '[0-9][0-9][0-9][0-9]-[0-9]?[0-9]-[0-9][0-9]'
```

2.6.6.1.19. *ITEM_UNITS_CONVERSION*

The ITEM_UNITS_CONVERSION category holds a table of conversion factors between the systems of units described in the ITEM_UNITS_LIST category. The systems of units are identified by a ***.from_code** and a ***.to_code**, which are both children of the item **_item_units_list.code**. The conversion is defined in terms of an arithmetic operator and a conversion factor, **_item_units_conversion.operator** and **_item_units_conversion.factor**, respectively.

2.6.6.1.20. *ITEM_UNITS_LIST*

The ITEM_UNITS_LIST category holds the descriptions of systems of physical units. The key item in this category is **_item_units_list.code**. Units are assigned to data items by references to this key from the ITEM_UNITS category.

2.6.6.2. DDL2 definitions describing categories

In this section, the DDL definitions that describe the properties of categories, category groups and subcategories are presented. Fig. 2.6.4.2 illustrates the organization of these categories.

2.6.6.2.1. *CATEGORY*

The category named CATEGORY contains the data items that describe the properties of collections of related data items. A DDL category is essentially a table. In this category the characteristics of the table as a whole are defined. This category includes the data items **_category.id** to identify a category name; **_category.description** to describe a category; **_category.mandatory_code** to indicate whether the category must appear in a data block; and **_category.implicit_key**, which can be used to merge like categories between data blocks. The category identifier **_category.id** is a component of the key in most of the DDL categories in this section. The parent definition of the category identifier and all its child relationships are defined in this category.

Because special rules exist in the STAR grammar for the specification of data items that belong to a common category, the organization of data items within categories has a significant influence on how these items may be expressed in a data file. For example, a data category may be specified only once within a STAR data block or save frame, and at any level of a STAR loop structure only data items of a common category may appear.

2.6.6.2.2. *CATEGORY_EXAMPLES*

The category named CATEGORY_EXAMPLES holds examples that apply to an entire category. This typically includes a complete specification of the category with annotations. An example specification consists of the text of the example, **_category_examples.case**, and an optional comment item, **_category_examples.detail**, which can be used to qualify the example. The key for this category includes the items **_category_examples.id** and **_category_examples.case**. The former is completely defined in the parent category named CATEGORY.

2.6.6.2.3. *CATEGORY_GROUP*

The category CATEGORY_GROUP names the category groups to which a category belongs. The assignment of a category to a category group is made when the category is defined. Each category group that is specified in this category must also be defined in the parent category, CATEGORY_GROUP_LIST. The basis for this category also includes the category identifier **_category_group.category_id**, which is completely defined in the parent category named CATEGORY.

2.6.6.2.4. *CATEGORY_GROUP_LIST*

The DDL category CATEGORY_GROUP_LIST holds data items that define category groups. Category groups are collections of related categories. Parent–child relationships may be defined for these groups. The specification of category groups and the relationships between these groups allow a complicated collection of categories to be organized into a hierarchy of more relevant groups. This higher level of structure is essential for large application dictionaries that may contain hundreds of category definitions.

The category CATEGORY_GROUP_LIST holds the description of each category group, **_category_group_list.description**, and an optional identifier of the parent group, **_category_group_list.parent_id**. Category groups can be formed from collections of base categories, those categories that hold data. Category groups can also be formed from collections of base categories and category groups.

Example 2.6.6.2 illustrates the category groups that are defined in this DDL. These include the group of categories that define categories, the group of categories defining data items and the group of categories that define properties of the dictionary. An additional compliance group is also defined for categories that are included specifically for compliance with previous versions of DDL. Each of these category groups is defined as a child of the group named **ddl_group** to which all of the base DDL categories belong.

2.6.6.2.5. *CATEGORY_KEY*

The category CATEGORY_KEY identifies the data items within a category that form the basis for the category. The category basis uniquely identifies each group or tuple of items in the category. In the analogy of the category as a table, no row in a table may have duplicate values for its key data items.

Example 2.6.6.2. *Category groups defined in the DDL2 dictionary.*

```
     loop_
     _category_group_list.id
     _category_group_list.parent_id
     _category_group_list.description

     'ddl_group'             .
;
     Component categories of the macromolecular DDL.
;
     'datablock_group'       'ddl_group'
;
     Categories that describe the characteristics of
     data blocks.
;
     'category_group'        'ddl_group'
;
     Categories that describe the characteristics of
     categories.
;
     'sub_category_group'    'ddl_group'
;
     Categories that describe the characteristics of
     subcategories.
;
     'item_group'            'ddl_group'
;
     Categories that describe the characteristics of
     data items.
;
     'dictionary_group'      'ddl_group'
;
     Categories that describe the dictionary.
;
     'compliance_group'      'ddl_group'
;
     Categories that are retained specifically for
     compliance with older versions of the DDL.
;
```

The choice of basis has important consequences in the specification of a category. It is important to ensure that the key items that form the category basis can unambiguously identify any tuple of data items within the category. If this is not the case, then it may not be possible to reliably recover data items that are stored in the category. Because key items are required to address each tuple of items in a category, key items are considered mandatory items in the category.

It is interesting to note how the key data items have been selected for the categories that define the DDL, and how this choice of key items influences the structure of the DDL dictionary. In the DDL category CATEGORY_KEY, the basis includes both the identifier for the category, **_category_key.id**, and the name of the key data item, **_category_key.name**. This choice of basis allows for any unique groups of items in a category to be defined as key items. Duplicate key-item values within a category are forbidden by the data model. In the DDL category ITEM_TYPE, the basis includes only the identifier for the item name, **_item_type.name**. This choice of basis has the desired effect of limiting the specification of item data type, **_item_type.code**, to a single choice for each data item.

2.6.6.2.6. *CATEGORY_METHODS*

The CATEGORY_METHODS category is used to associate method identifiers with categories. Any number of unique method identifiers may be associated with a category. The method identifiers reference the full method definitions in the parent METHOD_LIST category.

2.6.6.2.7. *SUB_CATEGORY*

The category SUB_CATEGORY provides data items to describe a subcategory and to associate a procedure with the subcategory (see Section 2.6.6.2.9). A subcategory is a set of data items within a category that have a particular association. A typical example would be a triad of positional coordinates x, y, z that are collectively assigned to a 'cartesian' subcategory.

2.6.6.2.8. *SUB_CATEGORY_EXAMPLES*

The DDL category SUB_CATEGORY_EXAMPLES holds examples of a subcategory. A subcategory example might illustrate valid instances of the items comprising the subcategory. An example specification contains the text of the example, `_sub_category_examples.case`, and an optional comment item, `_sub_category_examples.detail`, that can be used to qualify the example. The key for this category includes the items `_sub_category_examples.id` and `_sub_category_examples.case`. This compound basis permits multiple unique examples to be provided for each subcategory.

2.6.6.2.9. *SUB_CATEGORY_METHODS*

The SUB_CATEGORY_METHODS category is used to associate method identifiers with subcategories. Any number of unique method identifiers may be associated with a subcategory. The method identifiers reference the full method definitions in the parent METHOD_LIST category.

The procedure that is identified as `_sub_category_methods.method_id` may be used to validate the subcategory identified as `_sub_category_methods.sub_category_id`. Subcategory validation may be required in instances where conditions are placed on the values of data items within the subcategory that are more restrictive than those associated with each component data item. A simple example of such a restriction would be a normalization restriction on the components of a subcategory. Any procedure referenced in this category must also be defined in the category METHOD_LIST.

2.6.6.3. DDL2 definitions describing methods

In this section, the DDL categories that define the methods associated with data blocks, categories, subcategories and items are presented. Figs. 2.6.4.1, 2.6.4.2 and 2.6.4.3 illustrate the relationships between the method categories and other DDL categories.

2.6.6.3.1. *METHOD_LIST*

The METHOD_LIST category defines methods that can be associated with data blocks, categories, subcategories and items. This category attempts to capture only the essential information required to define these methods, without defining any implementation details. The implementation details are appropriately left to application-dictionary developers. It is assumed here that, within a domain of dictionaries, a consistent method interface will be adopted that is tailored to the requirements of that domain. This of course complicates the sharing of methods between domains; however, it would be impossible at this time to define an implementation strategy inside the DDL that would even begin to satisfy the diverse requirements of potential DDL users. Consequently, the definition of each method is limited to: its unique identifier, `_method_list.id`; a textual description, `_method_list.detail`; the source text of the method, `_method_list.inline`; the name of the language in which the method is expressed, `_method_list.language`; and a code to identify the purpose of the method, `_method_list.code`.

2.6.6.4. DDL2 definitions describing dictionaries and data blocks

In this section, the DDL categories that describe the characteristics of dictionaries and data blocks are presented. In this context, a dictionary is defined as a group of related definitions within a STAR data block. Fig. 2.6.4.3 illustrates the organization for these categories.

2.6.6.4.1. *DATABLOCK*

The DATABLOCK category holds the essential identifying information for a data block: the name of the data block, `_datablock.id`; and a description of the block, `_datablock.description`. The `_datablock.id` is the parent identifier for both `_category.implicit_key` and `_dictionary.datablock_id`. The former guarantees that the identifier for the data block, and hence the dictionary, is added implicitly to the key of each category.

2.6.6.4.2. *DATABLOCK_METHODS*

The DATABLOCK_METHODS category may be associated with a data block. The method identifiers reference the full method definitions in the parent METHOD_LIST category.

2.6.6.4.3. *DICTIONARY*

The DICTIONARY category holds the essential identifying information for a data dictionary. The items recorded in this category include the title for the dictionary, `_dictionary.title`, the current version identifier, `_dictionary.version`, and the data-block identifier in which the dictionary is defined, `_dictionary.datablock_id`. The version identifier references the parent identifier in the DICTIONARY_HISTORY category in which each dictionary revision is described.

2.6.6.4.4. *DICTIONARY_HISTORY*

The DICTIONARY_HISTORY category holds the revision history for a dictionary. Each revision is assigned a version identifier that acts as the key item for the category. Along with the version information, a text description of the revision and date of revision must be specified.

References

Allen, F. H., Barnard, J. M., Cook, A. F. P. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

Westbrook, J. D. & Hall, S. R. (1995). *A dictionary description language for macromolecular structure.* http://ndbserver.rutgers.edu/mmcif/ddl/.

3. CIF DATA DEFINITION AND CLASSIFICATION

BY H. M. BERMAN, H. J. BERNSTEIN, P. E. BOURNE, I. D. BROWN, P. M. D. FITZGERALD, S. R. HALL, K. HENRICK, B. MCMAHON, G. MADARIAGA, P. R. MALLINSON, B. H. TOBY, E. L. ULRICH, K. D. WATENPAUGH AND J. D. WESTBROOK

3.1. General considerations when defining a CIF data item (B. MCMAHON) 73
3.1.1. Introduction 73
3.1.2. Informal definition procedures 74
3.1.3. Formal definition process 74
3.1.4. Choice of data model 75
3.1.5. Constructing a DDL1 dictionary 76
3.1.6. Constructing a DDL2 dictionary 79
3.1.7. Composing new data definitions 83
3.1.8. Management of multiple dictionaries 85
3.1.9. Composite dictionaries 88
3.1.10. Public CIF dictionaries 89
References 91
3.2. Classification and use of core data (S. R. HALL, P. M. D. FITZGERALD AND B. MCMAHON) 92
3.2.1. Introduction 92
3.2.2. Experimental measurements 92
3.2.3. Analysis 98
3.2.4. Atomicity, chemistry and structure 102
3.2.5. Publication 112
3.2.6. File metadata 114
Appendix 3.2.1. Category structure of the core CIF dictionary 116
References 116
3.3. Classification and use of powder diffraction data (B. H. TOBY) 117
3.3.1. Introduction 117
3.3.2. Dictionary design considerations 117
3.3.3. pdCIF dictionary sections 117
3.3.4. Experimental measurements 118
3.3.5. Analysis 121
3.3.6. Atomicity, chemistry and structure 123
3.3.7. File metadata 123
3.3.8. pdCIF for storing unprocessed measurements 126
3.3.9. Use of pdCIF for Rietveld refinement results 128
3.3.10. Other pdCIF applications 129
Appendix 3.3.1. Category structure of the powder CIF dictionary 130
References 130

3.4. Classification and use of modulated and composite structures data (G. MADARIAGA) 131
3.4.1. Introduction 131
3.4.2. Dictionary design considerations 131
3.4.3. Arrangement of the dictionary 132
3.4.4. Use of the msCIF dictionary 137
Appendix 3.4.1. Category structure of the msCIF dictionary 139
References 139
3.5. Classification and use of electron density data (P. R. MALLINSON AND I. D. BROWN) 141
3.5.1. Introduction 141
3.5.2. Dictionary design considerations 141
3.5.3. Classification of data definitions 141
3.5.4. Development of the dictionary and supporting software 143
References 143
3.6. Classification and use of macromolecular data (P. M. D. FITZGERALD, J. D. WESTBROOK, P. E. BOURNE, B. MCMAHON, K. D. WATENPAUGH AND H. M. BERMAN) 144
3.6.1. Introduction 144
3.6.2. Considerations underlying the design of the dictionary 144
3.6.3. Overview of the mmCIF data model 145
3.6.4. Content of the macromolecular CIF dictionary 147
3.6.5. Experimental measurements 148
3.6.6. Analysis 152
3.6.7. Atomicity, chemistry and structure 164
3.6.8. Publication 190
3.6.9. File metadata 194
Appendix 3.6.1. Category structure of the mmCIF dictionary 195
Appendix 3.6.2. The Protein Data Bank exchange data dictionary (J. D. WESTBROOK, K. HENRICK, E. L. ULRICH AND H. M. BERMAN) 195
References 198
3.7. Classification and use of image data (H. J. BERNSTEIN) 199
3.7.1. Introduction 199
3.7.2. Binary image data 199
3.7.3. Axes 201
3.7.4. The diffraction experiment 202
Appendix 3.7.1. Category structure of the CBF/imgCIF dictionary 205
References 205
3.8. Classification and use of symmetry data (I. D. BROWN) 206
3.8.1. Introduction 206
3.8.2. Dictionary design considerations 206
3.8.3. Arrangement of the dictionary 207
3.8.4. Future developments 208
References 208

3.1. General considerations when defining a CIF data item

BY B. MCMAHON

3.1.1. Introduction

Much of the power and usefulness of the Crystallographic Information File (CIF) arises from the existence of a comprehensive set of data dictionaries that define all data items commonly used in the field. These are the dictionaries that are presented in Part 4 of this volume. The information contained in a CIF is expressed in terms of these data items. A data item consists of a value associated with a data name, or tag. The tag may appear immediately before a single data value or in the heading of a looped list where the values form a column. In either construction, the data value is identified by the tag and this unique character string is the key to the definition of the data value in the dictionary.

A data definition may include information such as a text description of the quantity, its physical units, the range within which valid values must lie, the names of other data items that are related by inheritance or derivation to the data item and so on. Placing this information in a dictionary file, rather than in the data file itself, has a number of important advantages. First, it encourages the standardization of unique tags for data items, which is an essential step towards the seamless and unambiguous exchange of information. Dictionaries also facilitate a globally accepted understanding of what each data item is, and thus ensure that different data files using the same tags have a consistent interpretation.

The existence of global dictionaries does not in any way restrict the expressive power of CIF. A CIF may contain items not in the standard dictionaries, as well as items in local dictionaries with quite idiosyncratic definitions. The choice of which items to include in a CIF depends on the capabilities of the applications that are intended to use the data in the file. It is also influenced by the extent to which the author of the file wishes the data to be retrievable without ambiguity in the future. Of course, the same applies to data in XML (Bray *et al.*, 1998; W3C, 2004) or other data languages. In the adoption and application of CIF as a specific exchange mechanism, the crystallographic community has imposed on itself a particular discipline: the strict definition of its data with carefully maintained dictionaries. This is not to be seen as a restriction but as a means to unambiguous and effective communication.

As mentioned above, data with local definitions are easily accommodated in a CIF. However, for a CIF to be an effective exchange medium, data definitions need to be accessible to the community of users. This is most efficient when commonly used data items are collected into a dictionary or dictionaries that are readily obtainable and centrally coordinated. This is why the CIF dictionaries, containing the definitions of standard data names and their attributes, are published and maintained by a technical committee of the International Union of Crystallography (IUCr): the Committee for the Maintenance of the CIF Standard (COMCIFS). The dictionaries employ a dictionary definition language or DDL (see Chapters 2.5 and 2.6) to describe relevant attributes of CIF data items.

Affiliation: BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

This chapter will discuss the general concepts behind defining data items in CIF dictionaries. It will describe how standard dictionaries may be constructed and disseminated, and also how local extensions may be built and used in ways that do not conflict with the need for community standards. Some necessary details about the administration of standard dictionaries are also provided.

3.1.1.1. Authorship of data dictionaries

A difficulty in developing a standard for information exchange across the field of crystallography is the breadth of the subject area and the many subdisciplines it includes. One feature of the construction of data dictionaries for CIF is the delegation of responsibility for identifying and defining the data items important within a research area to experts in that field. In consequence, a richer compilation of definitions results than would be possible from a single author or small group of authors. However, each subdiscipline will have its own emphases and requirements, and it becomes a challenge to accommodate the needs of each individual subdiscipline within the framework of the general body of definitions covering the entire subject area. COMCIFS deals with this challenge by initiating and ratifying dictionaries written by IUCr Commissions or other specialist groups.

3.1.1.2. Certification for community use

A further responsibility of COMCIFS is to try to harmonize the treatment of similar data requirements in different dictionaries and to maintain maximum compatibility between data files originating from different subdisciplines. To achieve this, COMCIFS can officially approve dictionaries submitted to and reviewed by it. It is these 'official' dictionaries that are included in this volume. Provisional dictionaries may also be issued and used within the relevant community before formal approval is given.

3.1.1.3. DDL versions

Ideally, compatibility between the data dictionaries originating from specific subdisciplines would be ensured by the adoption of the same attribute sets for data items. However, at this point in the evolution of the CIF standard, two slightly different attribute sets have become established. These are expressed in two versions of the dictionary definition language, DDL1 and DDL2 (detailed in Chapters 2.5 and 2.6, respectively). The differences arise because some subdisciplines benefit from a strict data model that is not appropriate in other areas. The core data items in crystallography must of course be accessible across the field, and so there are two formulations of the dictionary of core items, one in each DDL version. The existence of two formulations can make full information interchange across all areas of crystallography difficult, so work is under way to bring about a convergence of the two current representations (Hall *et al.*, 2002). It is particularly important for future interchange between crystallography and other related disciplines that a full understanding be reached of the best way to include different data structure models within a common interchange format.

In this chapter, there will be some discussion of the differences in practice between the DDL versions DDL1 and DDL2, as these will strongly influence the choice of formalism for a dictionary relevant to a subdiscipline not yet represented.

3.1.2. Informal definition procedures

Before considering the techniques for defining data items in standard globally adopted dictionaries, it is important to discuss the techniques for including information that is only of local interest in a way that does not conflict with public data names.

An author of a CIF is free to include data names for local use (*i.e.* names not intended for common use across the community). However, such local data names *must not* conflict with those defined in public dictionaries, since the data name alone identifies the meaning that one must attach to an associated data value. Some protocols and conventions exist to prevent conflict in data names when the local data name is invented or subsequently, when later public dictionaries are released.

An author may also define local data names in some completely informal manner; that is, there is no obligation to construct an attribute table in an external file that conforms to the style of the public dictionaries. Nevertheless, there are clear advantages to doing so: the author will benefit from standard software tools that validate data against dictionaries and the data names are more easily exported to the public domain if they subsequently become relevant to a wider community. In the following, it is assumed that the author of a new data name wishes to define fully its attributes in an appropriate standard dictionary formalism.

3.1.2.1. The _[local]_ prefix

The string `_[local]_` is *reserved* as a prefix to identify data names that do not appear in any public dictionary. (The left and right square brackets are included in this label.) Hence an author may construct private data names according to one of the following models, secure in the knowledge that the name will not appear in any global dictionary. With DDL1, a private data name will always have the form `_[local]_private_data_name`, while with DDL2 the forms `_[local]_new_category_name.private_data_name` and `_existing_category_name.[local]_private_data_name` may be used. The first DDL2 form is used for private data names in a category not already defined by a public dictionary; the second form permits the addition of local data names to an existing category. Note that the initial underscore character is dropped in the second DDL2 form.

While this convention guarantees that the new data name will not conflict with a public one, it cannot guarantee that it will not conflict with a local data name created by another author. Therefore these data names are appropriate only for testing purposes and not for release in data files that may be used by others.

3.1.2.2. Reserved prefixes

To guarantee that locally devised data names may be placed without name conflict in interchange data files, authors may register a reserved character string for their sole use. As with the special prefix `_[local]_` discussed in Section 3.1.2.1, the author's reserved prefix is simply an underscore-bounded string within the data name (*i.e.* it may not itself include an underscore character). For DDL1 applications it must be the first component of the data name; for DDL2 applications it forms the first component of the data name if describing data names in a category not defined in the official dictionaries; or the first component after the full stop (category delimiter) if the local data name is an extension to an existing category.

Table 3.1.2.1. *Reserved prefixes for private CIF data names*

String	Reserved for the use of
anbf	Australian National Beamline Facility
asd	Active Site Database
B+S	Software developers Bernstein + Sons
ccdc	Cambridge Crystallographic Data Centre
CCP4	*CCP*4 program system
cgraph	Oxford Cryosystems *Crystallographica* package
cifdic	Register of CIF dictionaries
crystmol	*CrystMol* package
csd	Cambridge Structural Database
ebi	European Bioinformatics Institute
edchem	Edinburgh University Chemistry Department
gsas	*GSAS* powder refinement system
gsk	Glaxo Smith Kline
iims	EBI project on integration of information about macromolecular structure
iucr	IUCr journal use
mdb	Model Database (Glaxo)
msd	EBI Molecular Structure Database Group
ndb	Nucleic Acids Database Project, Rutgers University
oxford	*CRYSTALS* package, University of Oxford
parvati	Validation and statistical summaries from *PARVATI* validation server
pdb	Protein Data Bank
pdbx	Protein Data Bank exchange dictionary
pdb2cif	Additions to mmCIF used by program *pdb2cif*
rcsb	Research Collaboratory for Structural Bioinformatics
shelx	*SHELXL* solution and refinement programs
vrf	Validation reply form (IUCr/*Acta Crystallographica* use)
wdc	Entries in the World Directory of Crystallographers
xtal	*Xtal* program system

Prefixes may be registered online through a web form at http://www.iucr.org/iucr-top/cif/spec/reserved.html. Table 3.1.2.1 gives a list of prefixes registered as of March 2005; this list will of course go out of date, but a current list will be maintained on the web at the address above.

An example of a data name incorporating a reserved prefix is the listing of a protein amino-acid sequence recorded temporarily by the Protein Data Bank before a protein structure is released, `_pdbx_prerelease_seq.seq_one_letter_code`.

3.1.2.3. Name spaces

The allocation of special prefixes as in Sections 3.1.2.1 and 3.1.2.2 above is a basic form of name-space allocation, because it gives authors the freedom to reproduce portions of otherwise standard data names within their own private constructions. This raises the wider question of whether a complete formalism for name-space allocation is needed. That is, the same data name might appear with different meanings in different files, provided it was clear which of the alternative definitions must be used in each case. For now, the decision has been taken not to permit the use of the same data names with different meanings in different contexts. This is to enforce uniformity of definition across the whole field of crystallography as far as is possible. This policy might be reviewed in the future if similar formalisms to CIF are created in related disciplines.

3.1.3. Formal definition process

This section describes the formal system for creating public dictionaries or appending to them. It includes information on the review and approval cycles currently required by COMCIFS, which could change if these procedures are modified. The IUCr web page

(http://www.iucr.org/iucr-top/cif) should be consulted for current practice. However, a short overview of the existing procedures is helpful in describing how the community can participate in extending the standard.

3.1.3.1. Dictionary maintenance groups

Each published dictionary authorized by COMCIFS has a group of specialists appointed or invited to extend and maintain the dictionary to serve the changing needs of the subdiscipline that sponsors the dictionary. Members of these dictionary maintenance groups (DMGs) may suggest extensions or corrigenda on their own initiative or may pass on requests for extensions from individual crystallographers. A DMG will typically debate and review any suggested amendments and produce a draft revised dictionary for approval by COMCIFS.

3.1.3.2. mmCIF review cycle

The macromolecular CIF dictionary covers a very broad and active field, and a more formal procedure exists for the submission and review of proposed extensions. Possible new definitions are submitted using *pro forma* dictionary templates to a member of an editorial board appointed by the mmCIF dictionary maintenance group. Accepted proposals are approved by the DMG and released for general community review in provisional extension dictionaries as circumstances require. The extension dictionary is revised as necessary and is finally incorporated within the parent mmCIF dictionary after COMCIFS approval has been granted.

3.1.3.3. New dictionaries

A completely new dictionary to cover a subdiscipline not otherwise catered for may be commissioned by COMCIFS or may arise from community action, occasionally sponsored by an IUCr Commission. A working group is appointed to create the dictionary and relevant example files or software. The working group is expected to test the new dictionary extensively within its own community before submitting it to COMCIFS for initial approval. It is the responsibility of COMCIFS to check the dictionary for technical consistency and for compatibility with related dictionaries. COMCIFS may refer the dictionary back to the working group for further revisions. When the dictionary finally receives formal COMCIFS approval and is published, a dictionary maintenance group is formed to promote its further development (Section 3.1.3.1). The DMG usually includes one or more members of the initial working group and at least one voting member of COMCIFS.

3.1.4. Choice of data model

The following sections of this chapter describe the technical considerations in defining data items within a dictionary. Fundamental to this is the *data model* on which the dictionary is based. The STAR File upon which CIF is based is a very versatile data format and can accommodate a variety of data models. However, the use within CIF of a single level of looping enforces a rather flat data structure and a typical CIF maps most easily onto a relational database model. This is implicit in DDL1, which assigns different attributes to data items depending on whether they appear in data loops or not. Generally speaking, one may consider a list header and its associated data values as the head and body of a table of data values. The list header (or equivalently the table head) identifies the data items ranged by column within the table. For the dictionary entries relating to the data names in the list header, the `_category` attribute collects together data items which may be looped together in the same table, and the `_list_reference`, `_list_mandatory` and `_list_uniqueness` attributes work together to indicate the data items that must be present and collectively have a unique value to identify a specific row in a table of values.

Example 3.1.4.1. *Core dictionary definitions for the atom-site labels and bond distances in a CIF table of molecular geometry.*

```
data_geom_bond_atom_site_label_
    loop_ _name
                        '_geom_bond_atom_site_label_1'
                        '_geom_bond_atom_site_label_2'
    _category                       geom_bond
    _type                           char
    _list                           yes
    _list_mandatory                 yes
    _list_link_parent             '_atom_site_label'
    _definition
;     The labels of two atom sites that form a bond.
      These must match labels specified as
      _atom_site_label in the atom list.
;

data_geom_bond_distance
    _name                         '_geom_bond_distance'
    _category                       geom_bond
    _type                           numb
    _type_conditions                esd
    _list                           yes
    _list_reference    '_geom_bond_atom_site_label_'
    _enumeration_range              0.0:
    _units                          A
    _units_detail                  'angstroms'
    _definition
;     The intramolecular bond distance in angstroms.
;
```

For example, the following example from the core CIF dictionary (Chapter 4.1) shows a table of bond distances. The dictionary definitions are given in Example 3.1.4.1.

```
loop_
_geom_bond_atom_site_label_1
_geom_bond_atom_site_label_2
_geom_bond_distance
  O1  C2   1.342(4)
  O1  C5   1.439(3)
  C2  C3   1.512(4)
  C2  O21  1.199(4)
  C3  N4   1.465(3)
  C3  C31  1.537(4)
  C3  H3   1.00(3)
  N4  C5   1.472(3)
```

Within the dictionary, entries for all of `_geom_bond_distance`, `_geom_bond_atom_site_label_1` and `_geom_bond_atom_site_label_2` share the same `_category` attribute, namely 'geom_bond'. (In the rest of this chapter, as elsewhere in the volume, we refer to categories by the upper-case form of their category attribute values; here, therefore, we are referring to the GEOM_BOND category.) The entry for `_geom_bond_distance` has a `_list_reference` value of `'_geom_bond_atom_site_label_'` indicating the data names that may be used to identify this particular table. The trailing underscore in this example indicates that all matching data names must be considered as components of a compound identifier; for this case the matching data names are `'_geom_bond_atom_site_label_1'` and `'_geom_bond_atom_site_label_2'`. The dictionary entry for `_geom_bond_atom_site_label_` has a `_list_mandatory` value of `yes`, indicating that these data items *must* be present within the table. In this way, the attributes specify the unique key within a database table (in this case, the key has multiple components: the labels of both contributing atom sites).

However, the mapping onto a relational database is not exact. In some cases CIFs may present data from a single category across

several tables, or the implied key may not have a unique value unless concatenated with other fields in the table row. For many applications this is only of academic interest; but in some subdisciplines it is important that the data model is constrained strictly to a relational one, and for those applications dictionaries built on the DDL2 formalism are more appropriate.

Of the dictionaries presented in this volume, the core, powder, modulated structures and electron density dictionaries use the DDL1 formalism and the symmetry, macromolecular and image dictionaries use the DDL2 formalism. The core dictionary uses DDL1 so that it can be used alongside other less rigorous dictionaries. The powder dictionary is one case of this, where the need to tabulate and merge extensive lists of raw or processed data is not well served by a relational model. Modulated structures are also best served by a data model that is not rigorously relational. The macromolecular dictionary uses DDL2 because many of the major database applications in macromolecular crystallography are relational in nature, but in consequence it contains a copy of the core data items re-expressed in DDL2 formalism. The image dictionary is in DDL2 because it was designed to operate closely alongside the macromolecular dictionary. The symmetry dictionary is an interesting case. It was constructed in DDL2 format as an exercise in supplying an extension dictionary immediately suitable for direct incorporation into other DDL2-based dictionaries and also suitable for transformation to the simpler DDL1 formalism as necessary to complement existing DDL1 dictionaries.

While the main difference between DDL1 and DDL2 lies in the rigour with which relational data structures are enforced, DDL2 also offers a larger set of attributes for specifying hierarchical relationships between data names and for typing data values, and in consequence a complete DDL2-based dictionary is richer (and correspondingly more complex to construct) than an equivalent DDL1 description.

There may be no obvious reason for selecting one formalism over the other when planning a new data dictionary, and prospective authors must give considerable thought to the merits of both formalisms. However, once the choice has been made, the structure of the dictionary and its component definitions is profoundly affected. The constructions of the two types of dictionary are discussed separately in Sections 3.1.5 and 3.1.6 below.

3.1.5. Constructing a DDL1 dictionary

Dictionaries constructed according to DDL1 have quite a simple structure. The structure is summarized in this section; Sections 3.1.5.1–3.1.5.4 provide more detail. Each definition is encapsulated within its own data block. Fig. 3.1.5.1 outlines the contents of the core CIF dictionary. The order of the data blocks has no significance, but it is common practice to start the file with the data block that describes the name, version and revision history of the dictionary itself and then to arrange data blocks in alphabetical order, sorted first on category then on names within a category. This practice is not always followed – for example, the powder dictionary is ordered by theme. The choice of order in a dictionary is only used for presentation and dictionary parsers should not assume or rely on any order of data blocks.

The name of a data block is usually constructed from the name of the data item it describes, *e.g.* `data_refln_phase_meas`. Where the data block describes an entire category instead of a single data item, the category name is followed by matching square brackets, which may contain an alphabetic code representing the dictionary name if it is an extension to the core dictionary (*e.g.* `data_refln_[]`, `data_audit_link_[ms]`). Where the data block defines several data names, the initial common portion of the names is used with a trailing underscore (*e.g.* `data_refln_`).

```
data_on_this_dictionary
    _dictionary_name              cif_core.dic
    _dictionary_version           2.3.1
    _dictionary_update            2005-06-27
    _dictionary_history
;
   1991-05-27  Created from CIF Dictionary text. SRH
   . . .
;
```

(*a*)

```
data_atom_site_[]
    _name                        '_atom_site_[]'
    _category                     category_overview
    _type                         null
    loop_ _example
    _example_detail   .   .

data_atom_site_adp_type
    _name                        '_atom_site_adp_type'
    _category                     atom_site
    _type                         char
    _definition                  'A standard code ...'

data_atom_site_aniso_B_
    loop_ _name                  '_atom_site_aniso_B_11'
                                  . . .
```

(*b*)

Fig. 3.1.5.1. Schematic structure of core CIF dictionary. (*a*) Dictionary identifiers. (*b*) Definitions of categories and data items.

A preliminary data block, by convention labelled with the header string `data_on_this_dictionary`, contains the dictionary identification information and revision history. The name of the dictionary itself (given by the data name `_dictionary_name`) is conventionally of the form `cif_`*`identifier`*`.dic`, where the *identifier* is a short code for the topic area of the dictionary (*e.g.* 'core' for the core dictionary, 'pd' for the powder dictionary, 'ms' for the modulated structures dictionary, 'rho' for the electron density dictionary).

Data names are classified by category. The `_category` attribute is a character string intended to indicate the 'natural grouping' of data items. If a data item occurs in a looped list, it must be grouped only with items from the same category. It is, however, permissible for a file to contain more than one looped list of the same category, provided that each loop has its own specific reference item identified by the `_list_reference` attribute of the data names included. Examples of this will be given below.

For each category, a data block is usually provided that contains information about the purpose of the category, generally illustrated with examples.

All other data blocks represent self-contained definitions of a single data item or a small set of closely related data items. The definition includes the physical units of and constraints on the values of the data labelled by the defined data name, and also information about relationships with other data items.

It is conventional, although not mandatory in DDL1 dictionaries, that the category name should appear as the leading component or components of a data name. For example, the data name `_exptl_crystal_colour` is a member of the core category EXPTL, while `_exptl_crystal_density_meas` is a member of the category EXPTL_CRYSTAL and `_exptl_crystal_face_perp_dist` is a member of the category EXPTL_CRYSTAL_FACE. However, it will

Example 3.1.5.1. *A DDL1 dictionary identification block.*

```
data_on_this_dictionary
    _dictionary_name            cif_core.dic
    _dictionary_version         2.3.1
    _dictionary_update          2005-06-27
    _dictionary_history
;   1991-05-27  Created from CIF Dictionary text. SRH
    1991-05-30  Validated with CYCLOPS & CIF ms. SRH
    ...
;
```

be seen that there is no sure way of working out the category from the complete data name except by referring to its **_category** attribute in the associated dictionary. This differs from the DDL2 convention of including an explicit separator (a full stop) between the category name and the remainder of a data name.

While it is not mandatory that a data name should incorporate its category name as a leading component, authors are strongly encouraged to adopt this convention. A small number of core data items that did not conform to this convention have been deprecated in later releases of the core dictionary. However, in the powder dictionary the convention has been broken so that one can present data sets separately or merge them together. In this dictionary, some data names beginning with the strings **_pd_calc**, **_pd_meas** and **_pd_proc** all belong formally to the category PD_DATA. This allows calculated data values to be tabulated with raw and processed measurements if this is useful.

One other case where a data name does not begin with its associated category name is that of the pseudo data names such as **_exptl_[]** that appear in the dictionary to describe the purpose of a category (Section 3.1.5.3). Such data names are always assigned the category CATEGORY_OVERVIEW and are further differentiated from other data names by having a data type of 'null'.

3.1.5.1. The dictionary identification block

As mentioned above, the dictionary file must contain information that unambiguously states its identity and version. In DDL1-based CIF dictionaries, this is achieved by itemizing the full set of dictionary attributes (see Section 2.5.6.5) within a data block named **data_on_this_dictionary**, as in Example 3.1.5.1 from the core dictionary.

3.1.5.2. Irreducible sets of data items

In general, a dictionary data block defines a single data item. However, there are instances where several related data names are defined in the same data block. Sometimes this has been done for convenience, to produce a compact listing of similar data names that have common attributes and whose small differences in meaning can best be expressed by a single definition. Such groupings are discouraged, except where they represent components of a larger entity that has no sensible meaning in the absence of any of the components. For example, the data block **data_refln_index_** defines the three data items **_refln_index_h**, **_refln_index_k** and **_refln_index_l** that represent the Miller indices of a reflection. All three indices must have a value in order to specify a reflection and so each has no meaning in isolation.

Note that there is no formal method of expressing this close relationship within DDL1 except by grouping the definitions in the same data block in this way. In DDL2 dictionaries, it is common to assign the components of an irreducible set to a specific subcategory.

Example 3.1.5.2. *A category description in a DDL1 dictionary.*

```
data_exptl_[]
    _name                         '_exptl_[]'
    _category                       category_overview
    _type                           null
    loop_ _example
    _example_detail
# - - - - - - - - - - - - - - - - - - - - - - - - - -
;   _exptl_absorpt_coefficient_mu     0.962
    _exptl_absorpt_correction_type    psi-scan
    _exptl_absorpt_process_details
                'North, Phillips & Mathews (1968)'
    _exptl_absorpt_correction_T_min   0.929
    _exptl_absorpt_correction_T_max   0.997
;
;   Example 1 - based on a paper by Steiner [Acta
                Cryst. (1996), C52, 2554-2556].
;
# - - - - - - - - - - - - - - - - - - - - - - - - - -
    _definition
;       Data items in the EXPTL category record
        details about the experimental work prior
        to the intensity measurements and details
        about the absorption-correction technique
        employed.
;
```

3.1.5.3. Category descriptions

As discussed above, categories in DDL1 are intended as 'natural groupings' of data items. To document the purpose of a category within a dictionary, 'pseudo' data names are used. All pseudo data names are assigned a **_category** attribute of category_overview and have an associated **_type** value of 'null'. They are also named by convention as ***_category_name_[dictionarycode]***, for example **_pd_data_[pd]** for the description of the PD_DATA category in the powder dictionary (indicated by the code 'pd' in square brackets). For the core dictionary, *dictionarycode* is not given, resulting in names like **_exptl_[]** to describe the EXPTL category.

Example 3.1.5.2 is a slightly edited extract from the core dictionary showing how a data block for a category description is composed, including the presence of an example.

Note that the *dictionarycode* extension allows a dictionary to include comments on items that it defines in a category already established in the core dictionary. For example, the modulated structures dictionary includes the category overview item **_audit_link_[ms]**. This describes the convention adopted to express the relationship between data blocks in a modulated structures data file using the **_audit_link_** data names already defined in the core dictionary.

3.1.5.4. Data-item definitions

The data blocks described in Sections 3.1.5.1 and 3.1.5.3 are used to identify the dictionary and to describe the nature and purpose of a category. The remaining data blocks in a dictionary provide the attributes of data values in a form suitable for machine extraction and validation. The following examples show how this is done for various types of data.

3.1.5.4.1. *Definitions of single quantities*

Example 3.1.5.3 is the core dictionary definition of the data name for the ambient temperature during the experiment. Because this is a single (non-looped) value, the relevant data name is one among several discrete items in the DIFFRN category. No further description of its relationship to other data items is required.

The type of the associated data value (*numb* for numerical) is specified, together with any constraint on its legal value. The range

Example 3.1.5.3. *A simple definition of a data item describing a physical quantity.*

```
data_diffrn_ambient_temperature
    _name                '_diffrn_ambient_temperature'
    _category                    diffrn
    _type                        numb
    _type_conditions             esd
    _enumeration_range           0.0:
    _units                       K
    _units_detail                kelvin
    _definition
;       The mean temperature in kelvins at which the
        intensities were measured.
;
```

specified (`0.0:`) indicates that it may be any non-negative real number. The physical units of the quantity are also indicated.

The `_definition` attribute is a concise human-readable documentation of the meaning associated with the data name.

Example 3.1.5.4 is taken from the powder dictionary and illustrates a data item that can have only one of a limited set of values. This data item indicates the geometry of the experiment. The associated data value is of type *char* and may legally take only one of the two possible values listed.

Example 3.1.5.4. *A data item that can take only one of a discrete set of allowed values.*

```
data_pd_spec_mount_mode
    _name                        '_pd_spec_mount_mode'
    _category                     pd_spec
    _type                         char
    loop_ _enumeration            reflection
                                  transmission
    _definition
;       A code describing the beam path through
        the specimen.
;
```

Example 3.1.5.5. *Definition relating a looped data item to the item used to identify a 'loop packet', or row of entries in a table.*

```
data_atom_site_U_iso_or_equiv
    _name               '_atom_site_U_iso_or_equiv'
    _category                    atom_site
    _type                        numb
    _type_conditions             esd
    _list                        yes
    _list_reference             '_atom_site_label'
```

Example 3.1.5.6. *Definition of a mandatory item within a loop.*

```
data_atom_site_label
    _name                        '_atom_site_label'
    _category                     atom_site
    _type                         char
    _list                         yes
    _list_mandatory               yes
    loop_ _list_link_child
          '_atom_site_aniso_label'
          '_geom_bond_atom_site_label_1'
          '_geom_bond_atom_site_label_2'
```

3.1.5.4.2. *Looped data*

Many of the attributes of looped data items, such as their physical units or valid numerical values, may be defined in exactly the same way as for non-looped data. However, more care needs to be taken to describe the relationships between different looped data items.

Consider the following example listing of some three-dimensional atom-site coordinates and displacement parameters.

```
loop_
     _atom_site_label
     _atom_site_fract_x
     _atom_site_fract_y
     _atom_site_fract_z
     _atom_site_U_iso_or_equiv
     _atom_site_thermal_displace_type
    O1 .4154(4) .56990(10) .3026000   .0600(10) Uani
    C2 .5630(5) .5087(2)   .32460(10) .060(2)   Uani
    C3 .5350(5) .4920(2)   .39970(10) .0480(10) Uani
    N4 .3570(3) .55580(10) .4167000   .0390(10) Uani
    C5 .3000(5) .6122(2)   .35810(10) .0450(10) Uani

loop_
     _atom_site_aniso_label
     _atom_site_aniso_U_11
     _atom_site_aniso_U_22
     _atom_site_aniso_U_33
     _atom_site_aniso_U_12
     _atom_site_aniso_U_13
     _atom_site_aniso_U_23
O1 .071(1) .076(1) .0342(9) .008(1)  .0051(9) -.0030(9)
C2 .060(2) .072(2) .047(1)  .002(2)  .013(1)  -.009(1)
C3 .038(1) .060(2) .044(1)  .007(1)  .001(1)  -.005(1)
N4 .037(1) .048(1) .0325(9) .0025(9) .0011(9) -.0011(9)
C5 .043(1) .060(1) .032(1)  .001(1) -.001(1)   .001(1)

loop_
     _geom_bond_atom_site_label_1
     _geom_bond_atom_site_label_2
     _geom_bond_distance
      O1  C2   1.342(4)
      O1  C5   1.439(3)
      C2  C3   1.512(4)
      C2  O21  1.199(4)
```

These loops, or tables of values, are properties of atom sites, each identified by a label such as O1. The definition of a data name such as `_atom_site_U_iso_or_equiv` expresses this by using the DDL1 `_list_reference` attribute (Example 3.1.5.5).

For an entry in the table to make sense, the site identifier must be present, so the definition for `_atom_site_label` declares it a mandatory item within its list (Example 3.1.5.6).

It is common for an atom-site identifier to be used in several related tabulations in a particular crystal structure description, and in a CIF description this means that it may occur in several different looped lists. The dictionary definition gives a formal account of this by listing the data names in other looped lists which are just different manifestations of this same item. This is done using the `_list_link_child` attribute, which identifies the data names to which the one being currently defined is 'parent'. In Example 3.1.5.6 (which is a subset of the full list in the core dictionary), `_atom_site_aniso_label`, `_geom_bond_atom_site_label_1` and `_geom_bond_atom_site_label_2` are identified as children of `_atom_site_label`.

It can been seen immediately that `_atom_site_aniso_label` is the atom-site identification label appearing in the second table in the example listing above, and the `_geom_bond_` items are clearly atom-site labels in a table of bonding properties between specified sites. There is, however, a difference between the two secondary tables: the bond-properties table is described by data items in the GEOM_BOND category, but the table of anisotropic displacement parameters includes data names that have the same `_category` attribute as the coordinate data items, namely ATOM_SITE. The latter is an example of multiple lists or tables belonging to the same category, a feature permitted only in DDL1-based data files.

3.1.5.4.3. *Units*

The physical units in which a quantitative data item must be expressed are identified by the DDL1 attributes `_units` and

_units_detail. The latter is a character field describing the units; the **_units** attribute is a code that may be interpreted by machine. In DDL1-based dictionaries, type codes are purely conventional, and there is no mechanism for converting units or relating quantities in different units. Table 3.1.5.1 lists the units codes used in the DDL1-based dictionaries described in this volume. There can be some inconsistencies: two codes ('s' and 'sec') are already in use to indicate the time unit of seconds.

The original CIF paper (Hall *et al.*, 1991) described a convention allowing physical quantities to be listed in a CIF in units other than those specified in the dictionary. Under this convention, a data name representing a value expressed in different units could be constructed by appending one of a series of known 'units extension codes' to the standard data name. Thus **_cell_length_a_pm** would represent a cell length expressed in picometres instead of the default ångströms. This approach is now deprecated, and all quantities must be expressed in the single unit permitted in their definition block. However, to allow the formal validation of old CIFs, a 'compatibility dictionary' is available which defines all data names that could have been constructed under this convention in a properly DDL1.4-compliant form. *This dictionary should only be used for validating old CIFs, and must not be used to construct new data files.* The dictionary is called cif_compat.dic in the IUCr CIF dictionary register (see Section 3.1.8.2).

Table 3.1.5.1. *Units codes and their interpretation in DDL1-based dictionaries*

Unit code (**_units**)	Meaning (**_units_detail**)
A	Ångströms
A^-1^	Reciprocal ångströms
A^2^	Ångströms squared
A^3^	Ångströms cubed
Da	Daltons
K	Kelvins
Kmin^-1^	Kelvins/minute
Mgm^-3^	Megagrams per cubic metre
\ms	Microseconds
deg	Degrees
deg/min	Degrees per minute
eV	Electronvolts
e_A^-3^	Electrons per cubic ångström
fm	Femtometres
kPa	Kilopascals
kV	Kilovolts
kW	Kilowatts
mA	Milliamperes
min	Minutes
mm	Millimetres
mm^-1^	Reciprocal millimetres
s	Seconds
sec	Seconds

3.1.6. Constructing a DDL2 dictionary

The DDL2 dictionary definition language was designed to specify a relational data model and has provision for including within a dictionary tables of relationships between data entries. Like a relational database which contains tables describing the data tables in the database, DDL2-based dictionaries contain definition blocks describing CIF categories, units and relationships as well as data items.

Unlike DDL1 dictionaries, a DDL2 dictionary is presented as a single data block. Within this data block a number of looped lists describe properties of the dictionary as a whole, or properties and relationships shared across the items defined in the dictionary. Typically these are: the dictionary name, version identifiers and revision history; the category groupings that give structure to the items defined by the dictionary; the labels that identify closely related data items; and the physical units employed in the dictionary, their definitions in terms of base units and their interconversion factors.

Definitions of individual data items and categories are contained within save frames. While the save frames are not referenced by name in any dictionary application, they permit multiple occurrences of data definition tags within the scope of a single data block and are therefore suitable for structuring a data dictionary. It is a convention that the name of a save frame defining a category is given in capitals, and the name of a save frame for a definition of a data item is given as lower-case. For example, **save_ATOM_SITE** is the name of the save frame defining the category with the **atom_site** identifier, while **save__atom_site.details** is the name of the save frame holding the definition of the individual data name **_atom_site.details** (note how the initial underscore character of the data name is preserved following the initial **save_** string of the save-frame name).

As with DDL1 dictionaries, the name of the dictionary itself (given by the data name **_dictionary.title**) is usually of the form **cif_*identifier*.dic**, where the *identifier* is a short code for the topic area of the dictionary (*e.g.* 'img' for the image dictionary, 'sym' for the symmetry dictionary).

As is invariable with DDL2 data names, the names themselves are formed from the category name separated by a full stop from the specific descriptor of the item.

Fig. 3.1.6.1 shows the structure of the macromolecular CIF dictionary. The ordering of the various looped lists and save frames is of no significance for machine parsing. The sole data block has the same name as the dictionary title string and the data block is introduced by the dictionary identification data items. The dictionary revision history introduces the file, followed by information about the extended data types and physical units used within the current dictionary. These are followed by the lists of closely related items (corresponding to 'irreducible sets' in DDL1 dictionaries and called 'subcategories' in the terminology of DDL2) and lists of category groupings. The body of the dictionary contains category and item definitions. Each category definition is followed by the definitions of its component data items. The ordering is alphabetic by category and then alphabetic by item name within categories.

3.1.6.1. Dictionary identification

Dictionary files must contain information that unambiguously states their identity and version. In DDL2-based dictionaries this is done using the dictionary attributes described in Section 2.6.6.4. The name of the data block comprising the whole content of a DDL2 dictionary is by convention the same as the dictionary identification string given as **_dictionary.title**. This value is repeated as the value of **_dictionary.datablock_id** (see Example 3.1.6.1) for use in checking the consistency of the dictionary.

The dictionary history is also an important audit record of changes to the dictionary content. Unlike in DDL1-based dictionaries where the history is contained in a single field, DDL2 provides a looped list of version labels, dates and annotations. For convenience, the history records in large DDL2-based dictionaries are sometimes placed at the end of the dictionary file.

3.1.6.2. Subcategory definitions

In the DDL1 formalism, particular relationships between data items may sometimes be stated within a text description or may be implied by the organization of the dictionary (where several data

```
data_mmcif_std.dic

    _dictionary.title           mmcif_std.dic
    _dictionary.version         2.0.09
    _dictionary.datablock_id    mmcif_std.dic
                          (a)

     loop_
         _dictionary_history.version
         _dictionary_history.update
         _dictionary_history.revision   .   .   .
                          (b)

     loop_
    _sub_category.id
    _sub_category.description    .   .

     loop_
    _category_group_list.id
    _category_group_list.parent_id
    _category_group_list.description   .   .   .
                          (c)

    loop_
    _item_type_list.code
    _item_type_list.primitive_code
    _item_type_list.construct
    _item_type_list.detail

    loop_
    _item_units_list.code
    _item_units_list.detail   .   .

     loop_
    _item_units_conversion.from_code
    _item_units_conversion.to_code
    _item_units_conversion.operator
    _item_units_conversion.factor   .   .   .   .
                          (d)


  save_CATEGORY_A   .   .   .   save_
    save__category_a.item_1   .   .   .   save_
    save__category_a.item_2   .   .   .   save_
    save__category_a.item_3   .   .   .   save_

  save_CATEGORY_B   .   .   .   save_
    save__category_b.item_1   .   .   .   save_
    save__category_b.item_2   .   .   .   save_
                          (e)
```

Fig. 3.1.6.1. Schematic structure of the macromolecular CIF dictionary. (*a*) Dictionary identifiers. (*b*) Dictionary history. (*c*) Subcategory and category group listings. (*d*) Data types, units descriptions and conversion tables. (*e*) Multiple category and item definition blocks.

items are defined in the same data block and are understood to share the common attributes itemized in that data block).

Within DDL2, there are mechanisms for more formal and machine-parsable statements of relationships. The `_sub_category.id` attribute is a label shared by several data items within a category that are related in a specific way described by the associated `_sub_category.description` attribute. The relationships may be rather general, such as elements of a matrix; or they may be specific physical properties or attributes, such as the collection of axis lengths of a unit cell. The dictionary should list all such labels that occur within its included data definition blocks. Example 3.1.6.2 is an extract from the macromolecular dictionary.

3.1.6.3. Category groupings

In the DDL2 data model, a *category* of data corresponds to a set of related data items that may be stored in a single relational database table. A number of such tables may collectively describe the complete properties of some physical object. This is expressed formally by assigning the same label (`_category_group.id`) to the relevant categories. While relationships between categories are implied in DDL1 dictionaries by the hierarchical structure of the names of data items, in DDL2 dictionaries the relationships are formally stated.

For subcategories, the category-group relationships present in the dictionary are listed in a separate looped list. Example 3.1.6.3 is an extract from the macromolecular dictionary. The `inclusive_group` entry shows the common parentage of all categories (and ultimately all data items) in the dictionary.

Example 3.1.6.1. *DDL2 dictionary identification entries.*

```
data_mmcif_std.dic

    _dictionary.title           mmcif_std.dic
    _dictionary.version         2.0.09
    _dictionary.datablock_id    mmcif_std.dic

     loop_
    _dictionary_history.version
    _dictionary_history.update
    _dictionary_history.revision
   0.1.1  1993-02-11
;  Highlighted all notes with # %%%%% surrounds.
;
   .  .  .
```

Example 3.1.6.2. *DDL2 subcategories defined in the mmCIF dictionary.*

```
loop_
    _sub_category.id
    _sub_category.description
    'fractional_coordinate'
; The collection of x, y, and z components of a
  position specified with reference to unit cell
  directions.
;
   'matrix'
; The collection of elements of a matrix.
;
   'miller_index'
; The collection of h, k, and l components of the
  Miller index of a reflection.
;
   'cell_length'
; The collection of a, b, and c axis lengths of a
  unit cell.
;
   'mm_atom_site_label'
; The collection of alt id, asym id, atom id, comp id
  and seq id components of the label for a
  macromolecular atom site.
;
```

3.1.6.4. Category definitions

In the DDL2 formalism, a category of data items may be mapped to a relational table. The dictionary entry for a category includes the name of the category (an identifying label which is referenced by the `_item.category_id` attribute of each component data item) and a list of the category groups of which it may be considered a member. The category *key* is explicitly specified – that is, the data item (or group of items) that uniquely identifies an individual row in a table of data of that category.

Where a category encompasses a set of data items that are not normally specified in a looped list, the category may nevertheless be taken to represent a degenerate table with a single row, and therefore there is still a category key. For degenerate categories the key value is often set equal to the name of the parent data block.

Example 3.1.6.3. *Category groups in a DDL2 dictionary.*

```
loop_
  _category_group_list.id
  _category_group_list.parent_id
  _category_group_list.description
    'inclusive_group'    .
; Categories that belong to the macromolecular
  dictionary.
;
    'atom_group'
    'inclusive_group'
; Categories that describe the properties of atoms.
;
    'audit_group'
    'inclusive_group'
; Categories that describe dictionary maintenance and
  identification.
;
    'cell_group'
    'inclusive_group'
; Categories that describe the unit cell.
;
```

Example 3.1.6.4. *A category description in a DDL2 dictionary.*

```
save_EXPTL
    _category.description
;   Data items in the EXPTL category record details
    about the experimental work prior to the
    intensity measurements and details about the
    absorption-correction technique employed.
;
    _category.id                    exptl
    _category.mandatory_code        no
    _category_key.name           '_exptl.entry_id'
     loop_
    _category_group.id             'inclusive_group'
                                   'exptl_group'
     loop_
    _category_examples.detail
    _category_examples.case
# - - - - - - - - - - - - - - - - - - - - - - - - - -
;  Example 1 - based on laboratory records for
   Yb(S-C5H4N)2 (THF)4
;
; _exptl.entry_id                     datablock1
  _exptl.absorpt_coefficient_mu       1.22
  _exptl.absorpt_correction_T_max     0.896
  _exptl.absorpt_correction_T_min     0.802
  _exptl.absorpt_correction_type      integration
  _exptl.absorpt_process_details
  ; Gaussian grid method from SHELX76
    Sheldrick, G. M., "SHELX-76: structure
    determination and refinement program",
    Cambridge University, UK, 1976
  ;
  _exptl.crystals_number              1
  _exptl.details
  ; Enraf-Nonius LT2 liquid nitrogen
    variable-temperature device used
  ;
  _exptl.method    'single-crystal x-ray diffraction'
  _exptl.method_details
  ; graphite monochromatized Cu K(alpha) fixed tube
    and Enraf-Nonius CAD4 diffractometer used
  ;
;
# - - - - - - - - - - - - - - - - - - - - - - - - - -
save_
```

Example 3.1.6.4 shows a category of non-looped core data items. It may be compared with the DDL1 version in Example 3.1.5.2.

For categories of looped items (those normally presented in a table of values) it is sometimes appropriate to have as the category key a data item that has the sole function of indexing unique table rows. However, it is also often the case that a composite key is formed from existing data items, and in these cases the category definition must loop the components of the key, as in Example 3.1.6.5 from the macromolecular dictionary definition of the GEOM_BOND category.

Example 3.1.6.5. *A DDL2 category with a composite key.*

```
save_GEOM_BOND
    _category.description
;   Data items in the GEOM_BOND category record
    details about the bond lengths as calculated
    from the contents of the ATOM, CELL and
    SYMMETRY data.
;
    _category.id                    geom_bond
    _category.mandatory_code        no
     loop_
    _category_key.name     '_geom_bond.atom_site_id_1'
                           '_geom_bond.atom_site_id_2'
                           '_geom_bond.site_symmetry_1'
                           '_geom_bond.site_symmetry_2'
     loop_
    _category_group.id             'inclusive_group'
                                   'geom_group'
     loop_
    _category_examples.detail
    _category_examples.case
# - - - - - - - - - - - - - - - - - - - - - - - - - -
;  Example 1 - based on data set TOZ of Willis,
   Beckwith & Tozer [Acta Cryst. (1991), C47,
   2276-2277].
;
;    loop_
     _geom_bond.atom_site_id_1
     _geom_bond.atom_site_id_2
     _geom_bond.dist
     _geom_bond.dist_esd
     _geom_bond.site_symmetry_1
     _geom_bond.site_symmetry_2
     _geom_bond.publ_flag
      O1  C2   1.342  0.004  1_555  1_555  yes
      O1  C5   1.439  0.003  1_555  1_555  yes
      C2  C3   1.512  0.004  1_555  1_555  yes
      C2  O21  1.199  0.004  1_555  1_555  yes
      C3  N4   1.465  0.003  1_555  1_555  yes
      C3  C31  1.537  0.004  1_555  1_555  yes
      C3  H3   1.00   0.03   1_555  1_555  ?
      N4  C5   1.472  0.003  1_555  1_555  yes
    # - - - - data truncated for brevity - - - -
;
# - - - - - - - - - - - - - - - - - - - - - - - - - -
save_
```

It must be remembered that, in practice, data files may lack some of the items required to determine the category key formally. For example, in the data set given in the GEOM_BOND example here, it is possible that the `_geom_bond.site_symmetry_` items may be absent because the listing is for a single connected molecule within an asymmetric unit. Robust parsing software must construct data keys by assigning NULL or other suitable default values to the missing key components.

Careful inspection of corresponding definitions in the DDL1 and DDL2 versions of core data items will demonstrate that the explicit category key specification in DDL2 dictionaries may be deduced within DDL1 dictionaries from the appropriate `_list_reference`, `_list_mandatory` and `_list_uniqueness` attributes of data-item definitions within a category (see also Section 2.5.6.4).

3.1.6.5. Data-item definitions

The bulk of a DDL2 data dictionary comprises the save frames that include descriptions of the meaning and properties of individual data names.

Unlike DDL1 dictionaries, where the definitions of several data names may be contained in a single data block (most commonly for a set of items that form a logical irreducible set), save frames in

Example 3.1.6.6. *Illustration of parent/child relationships between identifiers in related categories.*

```
loop_
  _struct_site.id
  _struct_site.details
    'P2 site C'
; residues with a contact < 3.7 Angstrom to an atom
  in the P2 moiety of the inhibitor in the
  conformation with _struct_asym.id = C
;
    'P2 site D'
; residues with a contact < 3.7 Angstrom to an atom
  in the P1 moiety of the inhibitor in the
  conformation with _struct_asym.id = D
;

loop_
  _struct_site_gen.id
  _struct_site_gen.site_id
  _struct_site_gen.label_comp_id
  _struct_site_gen.label_asym_id
  _struct_site_gen.label_seq_id
  _struct_site_gen.symmetry
  _struct_site_gen.details
    1  'P2 site C'  VAL  A   32  1_555  .
    2  'P2 site C'  ILE  A   47  1_555  .
    3  'P2 site C'  VAL  A   82  1_555  .
    4  'P2 site C'  ILE  A   84  1_555  .
    5  'P2 site D'  VAL  B  232  1_555  .
    6  'P2 site D'  ILE  B  247  1_555  .
    7  'P2 site D'  VAL  B  282  1_555  .
    8  'P2 site D'  ILE  B  284  1_555  .
```

Example 3.1.6.7. *A definition of an identifier which is parent to identifiers in other categories.*

```
save__struct_site.id
  _item_description.description
;     The value of _struct_site.id must uniquely
      identify a record in the STRUCT_SITE list.

      Note that this item need not be a number;
      it can be any unique identifier.
;
  loop_
    _item.name
    _item.category_id
    _item.mandatory_code
'_struct_site.id'            struct_site           yes
'_struct_site_gen.site_id'   struct_site_gen       yes
'_struct_site_keywords.site_id'
                             struct_site_keywords  yes
'_struct_site_view.site_id'  struct_site_view      yes
  loop_
    _item_linked.child_name
    _item_linked.parent_name
'_struct_site_gen.site_id'        '_struct_site.id'
'_struct_site_keywords.site_id'   '_struct_site.id'
'_struct_site_view.site_id'       '_struct_site.id'

  _item_type.code                 line
save_
```

DDL2 dictionaries each contain the definition for a single addressable concept.

For example, the three Miller index components of a diffraction reflection (**_diffrn_refln_index_h**, **_diffrn_refln_index_k**, **_diffrn_refln_index_l** that are described in the DDL1 core CIF dictionary in the data block **data_diffrn_refln_**) are described in a DDL2 dictionary in three separate save frames, **save__diffrn_refln.index_h**, **save__diffrn_refln.index_k** and **save__diffrn_refln.index_l**. In the DDL2 formalism, the intimate relationship between these three components is expressed through the common **_item_sub_category.id** value of `miller_index` and the mutual reference of the other Miller-index components by the **_item_dependent.dependent_name** entries in each separate save frame.

An apparent exception to this general rule is the case of save frames defining an item, often a category key, that is an identifier common to several categories. In this case, the save frame defining the 'parent' identifier implicitly defines the complete property set of each child identifier. For completeness, the respective child identifiers are each declared in their own save frames, but these act only as back references to the parent definition. This is explained more completely in Section 3.1.6.5.1 below.

3.1.6.5.1. *Inheritance of identifiers*

Example 3.1.6.6 is from an mmCIF of two related categories that describe characteristics of an active site in a macromolecular complex. The sites are described in general terms with a label and textual description in the STRUCT_SITE category (the first looped list in the example). Details of how each site is generated from a list of structural features form the STRUCT_SITE_GEN category (second loop or table).

It is clear that each instance of the data item **_struct_site_gen.site_id** in the second table must have one of the values listed as **_struct_site.id** in the first loop, because it is the purpose of these identifiers to relate the two sets of data: they are the glue between the two separate tables and must have the same values to ensure the referential integrity of the data set (that is, the consistency and completeness of cross-references between tables). Within a group of related categories like this, it is normal to consider one as the 'parent' and the others as 'children'.

Because all such linking data items must have compatible attributes, it is conventional in DDL2 dictionaries to define all the attributes in a single location, namely the save frame which hosts the definition of the 'parent' data item. In early drafts of DDL2 dictionaries, the 'children' were not referenced at all in separate save frames; software validating a data file against a dictionary was required to obtain all information about a child identifier from the contents of the save frame defining the parent. However, subsequent drafts introduced a minimal save frame for the children to accommodate dictionary browsers that depended on the existence of a separate definition block for each individual data item.

Consequently, the definition blocks in current DDL2 dictionaries conform to the structure in Example 3.1.6.7, which refers to the simple STRUCT_SITE example used above.

Note that the dependent data names are listed twice: once in the loop that declares their **_item.name** values and the categories with which they are associated; and again in a loop that makes the direction of the relationship explicit. A parent data item may have several children, but each child can have only a single parent (*i.e.* related data name whose value may be checked for referential integrity). Note also that each listed item has an **_item.mandatory_code** value of `yes`: because they are identifiers which link categories, they must be present in a table to allow the relationships between data items in different tables to be traced.

Other than the specific description text field, any declared attributes (in this example only the data type) have a common value across the set of related identifiers.

As mentioned above, it is not formally necessary to have a separate save frame for the individual children; but it is conventional to have such individual save frames containing minimal definitions that serve as back references to the primary information in the parent frame. These also provide somewhere for the specific text definitions for the children to be stored. The definition frame for **_struct_site_gen.id** is shown in Example 3.1.6.8.

Example 3.1.6.8. *Definition of a child identifier.*

```
save__struct_site_gen.id
  _item_description.description
;     The value of _struct_site_gen.id must uniquely
      identify a record in the STRUCT_SITE_GEN list.

      Note that this item need not be a number;
      it can be any unique identifier.
;
  _item.name                    '_struct_site_gen.id'
  _item.category_id               struct_site_gen
  _item.mandatory_code            yes
  _item_type.code                 line

save_
```

Example 3.1.6.9. *DDL2 definition of a physical quantity.*

```
save__diffrn.ambient_temp
    _item_description.description
;       The mean temperature in kelvins at which the
        intensities were measured.
;
    _item.name                   '_diffrn.ambient_temp'
    _item.category_id              diffrn
    _item.mandatory_code           no
    _item_aliases.alias_name
                          '_diffrn_ambient_temperature'
    _item_aliases.dictionary       cif_core.dic
    _item_aliases.version          2.0.1
     loop_
    _item_range.maximum
    _item_range.minimum             .    0.0
                                   0.0   0.0
    _item_related.related_name
                            '_diffrn.ambient_temp_esd'
    _item_related.function_code    associated_esd
    _item_type.code                float
    _item_type_conditions.code     esd
    _item_units.code               kelvins
save_
```

3.1.6.5.2. *Definitions of single quantities*

While it is important to ensure the referential integrity of the data in a CIF through proper book-keeping of links between tables, the crystallographer who wishes to create or extend a CIF dictionary will be more interested in the definitions of data items that refer to real physical quantities, the properties of a crystal or the details of the experiment. The DDL2 formalism makes it easy to create a detailed machine-readable listing of the attributes of such data.

Example 3.1.6.9 parallels the example chosen for DDL1 dictionaries of the ambient temperature during the experiment.

In the definition save frame, the category is specifically listed (although it is deducible from the DDL2 convention of separating the category name from the rest of the name by a full stop in the data name). The data type is specified as a floating-point number. (In the core dictionary there are fewer data types and the fact that the value may be a real rather than integer number must be inferred from the declared range.) The range of values is also specified with separate maximum and minimum values (unlike in DDL1 dictionaries, which give a single character string that must be parsed into its component minimum and maximum values). The assignment of the same value to a maximum and a minimum means that the absolute value is permitted; without the repeated '0.0' line the range in this example would be constrained to be positive definite; the equal value of 0.0 for maximum and minimum means that it may be identically zero.

The `_item_units.code` value must be one of the entries in the units table for the dictionary and can thus be converted into other units as specified in the units conversion table.

The aliases entries identify the corresponding quantity defined in the DDL1 core dictionary.

3.1.6.6. Units

As with data files described by DDL1 dictionaries, the physical unit associated with a quantitative value in a DDL2-based file is specified in the relevant dictionary. There is no option to express the quantity in other units. However, DDL2 permits a dictionary file to store not only a table of the units referred to in the dictionary (listed under `_item_units_list.code` and the accompanying descriptive item `_item_units_list.detail`), but also a table specifying the conversion factors between individual codes in the `_item_units_list.code` list. In principle, this allows a program to combine or otherwise manipulate different physical quantities while handling the units properly.

3.1.7. Composing new data definitions

Preceding sections have described the framework within which CIF dictionaries exist and are used, and their individual formal structures. While this is important for presenting the definition of new data items, it does not address what is often the most difficult question: what quantities, concepts or relationships merit separate data items? On the one hand, the extensibility of CIF provides great freedom of choice: anything that can be characterized as a separate idea may be assigned a new data name and set of attributes. On the other hand, there are practical constraints on designing software to write and read a format that is boundless in principle, and some care must be taken to organize new definitions economically and in an ordered way.

3.1.7.1. Granularity

Perhaps the most obvious decision that needs to be made is the level of detail or granularity chosen to describe the topic of interest. CIF data items may be very specific (the deadtime in microseconds of the detector used to measure diffraction intensities in an experiment) or very general (the text of a scientific paper). In general, a data name should correspond to a single well defined quantity or concept within the area of interest of a particular application. It can be seen that the level of granularity is determined by the requirements of the end application.

A practical example of determining an appropriate level of granularity is given by the core dictionary definitions for bibliographic references cited in a CIF. The dictionary originally contained a single character field, `_publ_section_references`, which was intended to contain the complete reference list for an article as undifferentiated text. *Notes for Authors* in journals accepting articles in CIF format advised authors to separate the references within the field with blank lines, but otherwise no structure was imposed upon the field. In a subsequent revision to the core dictionary, the much richer CITATION category was introduced to allow the structured presentation of references to journal articles and chapters of books. This was intended to aid queries to bibliographic databases. However, a full structured markup of references with multiple authors or editors in CIF requires additional categories, so that the details of the reference may be spread across three tables corresponding to the CITATION, CITATION_AUTHOR and CITATION_EDITOR categories. Populating several disjoint tables greatly complicates the author's task of writing a reference list. Moreover, the CITATION category does not yet cover all the many different types of bibliographic reference that it is possible to specify, and is therefore suitable only for references to journal articles and chapters of books. However, it is pos-

sible to write a program that can deduce the structure of a standard reference within an undifferentiated reference list (provided the journal guidelines have been followed by the author) to the extent that enough information can be extracted to add hyperlinks to references using a cross-publisher reference linking service such as CrossRef (CrossRef, 2004). Therefore, in practice, IUCr journals still ask the author of an article to supply their reference list in the `_publ_section_references` field, rather than using the apparently more useful `_citation_` fields. It remains to be seen whether this is the best strategy in the long term.

In more technical topic areas, the details of an experimental instrument could be described by a huge number of possible data names, ranging from the manufacturer's serial number to the colour of the instrument casing. However, many of these details are irrelevant to the analysis of the data generated by the instrument, so the characteristics of an instrument that are assigned individual data names are typically just those parameters that need to be entered in equations describing the calibration or interpretation of the data it generates.

3.1.7.2. Category 'special details' fields

When the specific items in a particular topic area that need to be recorded under their own data names have been decided, there is likely to be other information that could be recorded, but is felt to be irrelevant to the immediate purposes of the data collection and analysis. It is good practice to provide a place in the CIF for such additional information; it encourages an author to record the infomation and permits data mining at a later stage. Each category typically contains a data name with the suffix `_details` (or `_special_details`) which identifies a text field in which additional information relating to the category may be stored. This field often contains explanatory text qualifying the information recorded elsewhere in the same category, but it might contain additional specific items of information for which no data name is given and for which no obvious application is envisaged. This helps to guard against the loss of information that might be put to good use in the future. Of course, if a `*_details` field is regularly used to store some specific item of information *and* this information is seen to be valuable in the analysis or interpretation of data elsewhere in the file, there is a case for defining a new, separate tag for this information.

3.1.7.3. Construction of data names

Since a dictionary definition contains all the machine-readable attributes necessary for validating the contents of a data field, the data name itself may be an arbitrary tag, devoid of semantic content. However, while dictionary-driven access to a CIF is useful in many cases, there are circumstances where it is useful to browse the file. It is therefore helpful to construct a data name in a way that gives a good indication of the quantity described. From the beginning, CIF data names have been constructed from self-descriptive components in an order that reflects the hierarchical relationship of the component ideas, from highest (most general) level to lowest (most specific) level when read from left to right.

In a typical example from the core CIF dictionary, the data name `_atom_site_type_symbol` defines a code (`symbol`) indicating the chemical nature (`type`) of the occupant of a location in the crystal lattice (`atom_site`). The equivalent data name from the mmCIF dictionary, `_atom_site.type_symbol`, explicitly separates the category to which the data name belongs from its more specific qualifiers by using a full stop (`.`) instead of an underscore (_). While this use of a full stop is mandated in DDL2 dictionaries, it should

```
_database_code_CSD                 'VOBYUG'
                        (a)

_database_2.database_id            'PDB'
_database_2.database_code          '5HVP'
                        (b)
```

Fig. 3.1.7.1. Alternative quantities described (*a*) by data-name extension (core dictionary) or (*b*) by paired data names (mmCIF dictionary).

nevertheless be considered a convenience, since the category membership is explicitly listed in the dictionary definition frame for every data name.

However, it may not always be easy to establish the best order of components when constructing a new data name. In the JOURNAL category, there was initially some uncertainty about whether to associate the telephone numbers of different contact persons by appending codes such as `_coeditor` and `_techeditor` to a common base name. In the end, the order of components was reversed to give names like `_journal_coeditor_phone` and `_journal_techeditor_phone`. Examining the JOURNAL category in the core CIF dictionary will show why this was done. Similarly, the extension of geometry categories to include details of hydrogen bonding went through a stage of discussing adding new data names to the existing categories, but with suffixes indicating that the components were participating in hydrogen bonding, before it was decided that a completely new category for describing all elements of a hydrogen bond was justified. These examples show that the correct ordering of components within a data name is closely related to the perceived classification of data names by category and subcategory.

Sometimes it is useful to differentiate alternative data items by appending a suffix to a root data name. For example, the core dictionary defines several data names for recording the reference codes associated with a data block by different databases: `_database_code_CAS`, `_database_code_CSD` *etc.* This is convenient where there are two or three alternatives, but becomes unwieldy when the number of possibilities increases, because new data names need to be defined for each new alternative case. A better solution is to have a single base name and a companion data item that defines which of the available alternatives the base item refers to. The mmCIF dictionary follows this principle: the category DATABASE_2 contains two data names, `_database_2.database_code` (the value of which is an assigned database code) and `_database_2.database_id` (the value of which identifies which of the possible databases assigned the code) (Fig. 3.1.7.1).

Note the distinction between a data name constructed with a suffix indicating a particular database, and a data name which incorporates a prefix registered for the private use of a database. The data name `_database_code_PDB` is a *public* data name specifying an entry in the Protein Data Bank, while `_pdb_database_code` is a *private* data name used for some internal purpose by the Protein Data Bank (see Section 3.1.8.2).

3.1.7.4. Parsable data values *versus* separate data names

An advantage of defining multiple data names for the individual components of a complicated quantity is that there is no ambiguity in resolving the separate components. Hence the Miller indices of a reflection in the list of diffraction measurements are specified in the core dictionary by the group of three data names `_diffrn_refln_index_h`, `_diffrn_refln_index_k` and `_diffrn_refln_index_l`. In principle, a single data name

associated with the group of three values in some well defined format (*e.g.* comma separated, as h, k, l) could have been defined instead. However, this would require a parser to understand the internal structure of the value so that it could parse out the separate values for h, k and l.

On the other hand, there are many examples of data values that are stored as string values parsable into distinct components. An extreme example is the reference list mentioned in Section 3.1.7.1. More common are dates (`_audit_creation_date`), chemical formulae (*e.g.* `_chemical_formula_moiety`), symmetry operations (`_symmetry_equiv_pos_as_xyz`) or symmetry transformation codes (`_geom_bond_site_symmetry_1`). There is no definitive answer as to which approach is preferred in a specific case. In general, the separation of the components of a compound value is preferred when a known application will make use of the separate components individually. For instance, applications may list structure factors according to a number of ordering conventions on individual Miller indices. As an extreme example of separating the components of a compound value, the mmCIF dictionary defines data names for the standard uncertainty values of most of the measurable quantities it describes, while the core dictionary just uses the convention that a standard uncertainty is specified by appending an integer in parentheses to a numeric value.

When compound values are left as parsable strings, the parsing rules for individual data items need to be made known to applications. The DDL1 attribute `_type_construct` was envisaged as a mechanism for representing the components of a data value with a combination of regular expressions and reference to primitive data items, but this has not been implemented in existing CIF dictionaries (or in dictionary utility software). An alternative approach used in DDL2-based dictionaries defines within the dictionaries a number of extended data types (expressed in regular-expression notation through the attribute `_item_type_list.code`).

A related problem is how to handle data names that describe an indeterminate number of parameters. For example, in the modulated structures dictionary an extra eight Miller indices are defined to span a reciprocal space of dimension up to 11. In principle, the dimensionality could be extended without limit. According to the practice of defining a unique data name for each modulation dimension, new data names would need to be defined as required to describe higher-dimensional systems. Beyond a certain point this will become unwieldy, as will the set of data names required to describe the n^2 components of the W matrix for a modulated structure of dimensionality n (`_cell_subsystem_matrix_W_1_1` *etc.*).

The modulated structures dictionary was constrained to define extended Miller indices in this way for compatibility with the core dictionary. Data names describing new quantities that are subject to similar unbounded extensibility should perhaps refer to values that are parsable into vector or matrix components of arbitrary dimension.

3.1.7.5. Consistency of abbreviations

One further consideration when constructing a data name is the use of consistent abbreviations within the components of the data name. This is of course a matter of style, since if a data name is fully defined in a dictionary with a machine-readable attribute set, the data name itself can be anything. Nonetheless, to help to find and group similar data names it is best to avoid too many different abbreviations.

Table 3.1.7.1 lists the abbreviations used in the current public dictionaries. Note that there are already cases where different abbreviations are used for the same term.

3.1.8. Management of multiple dictionaries

So far this chapter has discussed the mechanics of writing dictionary definitions and of assembling a collection of definitions in a single global or local dictionary file. In practice, the set of data names in a CIF data file may include names defined in several dictionary files. A mechanism is required to identify and locate the dictionaries relevant to an individual data file. In addition, because dictionaries are suitable for automated validation of the contents of a data file, it is convenient to be able to overlay the attributes listed in a dictionary with an alternative set that permit validation against modified local criteria. This section describes protocols for identifying, locating and overlaying dictionary files and fragments of dictionary files.

3.1.8.1. Identification of dictionaries relevant to a data file

A CIF data file should declare within each of its data blocks the names, version numbers and, where appropriate, locations of the global and local dictionaries that contain definitions of the data names used in that block. For DDL1 dictionaries, the relevant identifiers are the items `_audit_conform_dict_name`, `_audit_conform_dict_version` and `_audit_conform_dict_location`, defined in the core dictionary. DDL2 dictionaries are identified by the equivalent items `_audit_conform.dict_name`, `*.dict_version` and `*.dict_location`. For convenience, the DDL1 versions will be used in the following discussion.

The values of the items `_audit_conform_dict_name` and `_audit_conform_dict_version` are character strings that match the values of the `_dictionary_name` and `_dictionary_version` identifiers in the dictionary that defines the relevant data names. Validation against the latest version of a dictionary should always be sufficient, since every effort is made to ensure that a dictionary evolves only by extension, not by revising or removing parts of previous versions of the dictionary. Nevertheless, including `_audit_conform_dict_version` is encouraged: it can be useful to confirm which version of the dictionary the CIF was initially validated against.

The data item `_audit_conform_dict_location` may be used to specify a file name or uniform resource locator (URL). However, a file name on a single computer or network will be of use only to an application with the same view of the local file system, and so is not portable. A URL may be a better indicator of the location of a dictionary file on the Internet, but can go out of date as server names, addresses and file-system organization change over time. The preferred method for locating a dictionary file is to make use of a dynamic register, as described in Section 3.1.8.2. Nevertheless, `_audit_conform_dict_location` remains a valid data item that may be of legitimate use, particularly in managing local applications.

The following example demonstrates a statement of dictionary conformance in a data file describing a powder diffraction experiment with some additional local data items:

```
loop_
  _audit_conform_dict_name
  _audit_conform_dict_version
  _audit_conform_dict_location
    cif_core.dic      2.3.1   .
    cif_pd.dic        1.0.1   .
    cif_local_my.dic  1.0
       /usr/local/dics/my_local_dictionary
```

Table 3.1.7.1. *Abbreviations in CIF data names*

Terms for which abbreviations are defined are sometimes found unabbreviated.

Abbreviation	Term	Abbreviation	Term	Abbreviation	Term
abbrev	abbreviation	eqn	equation	oper	operation
abs	absolute (configuration, not structure)	esd	standard uncertainty (estimated standard deviation) (*see* su)	org	organism
absorpt	absorption			orient	orientation
alt	alternative	expt	experiment	origx	orthogonal coordinate matrix (PDB files)
amp	amplitude	exptl	experimental	os	operating system
AN	accession number	fom	figure of merit	param	parameter
anal	analyser	fract	fractional	pd	powder diffraction
aniso	anisotropic*	Fsqd	*F* squared	PDB	Protein Data Bank
anisotrop	anisotropic*	gen	generation	PDF	Powder Diffraction File
anom	anomalous	gen	generator	perp	perpendicular
ASTM	American Society for Testing and Materials	gen	genetic	phos	phosphate
asym	asymmetric	geom	geometric	pk	peak
atten	attenuation	H-M	Hermann–Mauguin	polarisn	polarization
au	arbitrary units	ha	heavy atom	poly	polymer
auth	author	hbond	hydrogen bond	pos	position
av	average	hist	history	prep	preparation
ax	axial	horiz	horizontal	proc	processed
B	*B* form of atomic displacement parameter (a.d.p.)	I	intensity	prof	profile
		ICSD	Inorganic Crystal Structure Database	prot	protein
backgd	background*	id	identifier	ptnr	partner
beg	begin	illum	illumination	publ	publication
bg	background*	imag	imaginary	R	agreement index
biol	biology	inc	increment	rad	radius
bkg	background*	incl	include	recd	received
bond	bonding	info	information	recip	reciprocal
Bsol	*B* form of a.d.p. for solvent	instr	instrument	ref	reference
calc	calculated	Int	international	refine	refinement
calib	calibration (pd)	ISBN	International Standard Book Number	refln	reflection
cartn	Cartesian	iso	isotropic	reflns	reflections
CAS	*Chemical Abstracts* Service	iso	isomorphous	res	resolution
char	characterization (pd)	ISSN	International Standard Serial Number	restr	restraints
chem	chemical	IUCr	International Union of Crystallography	rev	revision
chir	chirality	IUPAC	International Union of Pure and Applied Chemistry	Rmerge	agreement index of merging
clust	cluster			rms	root mean square
coef	coefficient	len	length	rot	rotation
com	common	lim	limit	S	goodness of fit
comp	component	loc	lack of closure	samp	sample
conc	concentration	ls	least squares	scat	scattering factor
conf	conformation	max	maximum	seq	sequence
config	configuration	MDF	Metals Data File	sigI	$\sigma(I)$*
conform	conformant	meanI	mean intensity	sigmaI	$\sigma(I)$*
conn	connectivity	meas	measured	sint	$\sin\theta$
cons	constant	mid	middle (between max and min)	sint/lambda	$\sin(\theta)/\lambda$*
CSD	Cambridge Structural Database	min	minimum	sol	solvent
db	database	mod	modification	spec	specimen
defn	definition	mods	modifications	src	source
detc	detector	mon	monomer	std	standard
der	derivative	monochr	monochromator (pd)*	stol	$\sin(\theta)/\lambda$*
dev	standard deviation	mono	monochromator (pd)*	struct	structure
dict	dictionary	nat	natural	su	standard uncertainty
dif	difference*	NBS	National Bureau of Standards (now National Institute of Standards and Technology)	suppl	supplementary
diff	difference*			sys	systematic
diffr	diffractometer			tbar	mean path length
diffrn	diffraction	NCA	number of connected atoms	temp	temperature
displace	displacement	ncs	noncrystallographic symmetry	tor	torsion angle
dist	distance	netI	net intensity	tran	transformation*
divg	divergence	NH	number of connected hydrogen atoms	transf	transformation*
dom	domain	nha	non-hydrogen atoms	transform	transformation*
dtime	deadtime	norm	normal	tvect	translation vector (PDB files)
ens	ensemble	nst	nonstandard	vert	vertical
eq	equatorial*	nucl	nucleic acid	wR	weighted agreement index
equat	equatorial*	num	number	wt	weight
equiv	equivalent	obs	observed		

* Terms with multiple definitions.

It is clear that the location specified for the local dictionary is only meaningful for applications running on the same computer or network, and therefore the ability to validate against this local dictionary is not portable. On the other hand, it may be that the local data names used by the authors of this CIF are not intended to have meaning outside their own laboratory.

3.1.8.2. The dictionary register

COMCIFS maintains a register of dictionaries known to it, including the identifying name and version strings within those dictionaries. The register also includes the location of each dictionary, expressed at present as a URL designed to allow retrieval by file transfer protocol (ftp) from the IUCr server. Changes in the location of a particular dictionary file can be made by modifying the entry in the register, avoiding the problem of specifying a URL in a data file that would then become outdated if the dictionary was moved. Dictionary applications can consult the register (according to a protocol outlined below) to locate and retrieve the dictionaries needed for validating data files. It is of course essential that the validation software knows how to locate the register. The location is at present given by the URL ftp://ftp.iucr.org/pub/cifdics/cifdic.register.

The problem of changing URLs has therefore not disappeared completely, but is at least confined to the need to maintain one single address.

Table 3.1.8.1. *CIF dictionary register (maintained as a STAR File)*

```
data_validation_dictionaries
  loop_
    _cifdic_dictionary.name
    _cifdic_dictionary.version
    _cifdic_dictionary.DDL_compliance
    _cifdic_dictionary.reserved_prefix
    _cifdic_dictionary.URL
    _cifdic_dictionary.description
  cif_core.dic     .     1.4.1    .
      ftp://ftp.iucr.org/pub/cifdics/cif_core.dic
      'Core CIF Dictionary'
  cif_core.dic   1.0    .    .
      ftp://ftp.iucr.org/pub/cifdics/cifdic.C91
      'Original Core CIF Dictionary'
  cif_core.dic  2.3.1  1.4.1    .
      ftp://ftp.iucr.org/pub/cifdics/cif_core_2.3.1.dic
      'Core CIF Dictionary'
  cif_pd.dic       .      1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_pd.dic
      'Powder CIF Dictionary'
  cif_pd.dic     1.0.1     1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_pd_1.0.1.dic
      'Powder CIF Dictionary'
  cif_ms.dic       .      1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_ms.dic
      'Modulated structures CIF Dictionary'
  cif_ms.dic     1.0.1     1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_ms_1.0.1.dic
      'Modulated structures CIF Dictionary'
  cif_rho.dic      .      1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_rho.dic
      'Modulated structures CIF Dictionary'
  cif_rho.dic     1.0.1     1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_rho_1.0.1.dic
      'Electron density CIF Dictionary'
  cif_mm.dic      .     2.1.2   .
      ftp://ftp.iucr.org/pub/cifdics/cif_mm.dic
      'Macromolecular CIF Dictionary'
  cif_mm.dic     1.0    2.1.2   .
      ftp://ftp.iucr.org/pub/cifdics/cif_mm_1.0.dic
      'Macromolecular CIF Dictionary'
  mmcif_std.dic     .    2.1.6   .
      ftp://ftp.iucr.org/pub/cifdics/mmcif_std.dic
      'Macromolecular CIF Dictionary'
  mmcif_std.dic     2.0.09    2.1.6   .
      ftp://ftp.iucr.org/pub/cifdics/cif_mm_2.0.09.dic
      'Macromolecular CIF Dictionary'
  cif_img.dic      .     2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_img.dic
      'Image CIF Dictionary'
  cif_img.dic     1.0    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_img_1.0.dic
      'Image CIF Dictionary'
  cif_img.dic     1.3.2    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_img_1.3.2.dic
      'Image CIF Dictionary'
  cif_sym.dic     .     2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_sym.dic
      'Symmetry CIF Dictionary'
  cif_sym.dic    1.0.1    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_sym_1.0.1.dic
      'Symmetry CIF Dictionary'
  cif_compat.dic    .    1.4   .
      ftp://ftp.iucr.org/pub/cifdics/cif_compat.dic
      'Legacy CIF Dictionary of deprecated terms'
  ddl_core.dic    .    1.4.1   .
      ftp://ftp.iucr.org/pub/cifdics/ddl_core.dic
      'Non-relational dictionary definition language'
  ddl_core_2.1.3.dic    .    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/ddl_core_2.1.3.dic
      'Relational dictionary definition language'
  mmcif_ddl.dic    .    2.1.6   .
      ftp://ftp.iucr.org/pub/cifdics/mmcif_ddl.dic
      'Relational dictionary definition language'
```

Table 3.1.8.1 is an extract from the current register. The latest version of the register will always be available from the URL given above.

The entries for each dictionary include one with the version string set to '.', representing the current version; this is the version that should be retrieved unless a data file specifies otherwise.

Note that the register may also contain locators for local dictionaries constructed by owners of reserved prefixes (Section 3.1.2.2) when the owner has requested that a dictionary of local names be made publicly available. An appropriate name for a local dictionary in the register (**_dictionary_name** or **_dictionary.title** for DDL1 or DDL2 dictionaries, respectively) would be **cif_local_*myprefix*.dic**, where the string indicated by *myprefix* is one of the prefixes reserved for private use by the author of the dictionary (see Section 3.1.2.2). This scheme complements the naming convention for public dictionaries.

3.1.8.3. Locating a dictionary for validation

The following protocol applies to the creation and use of software designed to locate the dictionaries referenced by a data file and validate the data file against them. The protocol is necessary to address the issues that arise because dictionaries evolve through various audited versions, because not all dictionaries referenced by a data file may be accessible, and because data files might not in practice contain pointers to their associated dictionaries.

Software source code for applications that use CIF dictionaries to validate the contents of data files should be distributed with a copy of the most recent version of the register of dictionaries, and with the URL of the master copy hard-coded. Library utilities should be provided that permit local cacheing of the register file and the ability to download and replace the cached register at regular intervals. Individual dictionary files located and retrieved through the use of the register should also be cached locally, to guard against temporary unavailability of network resources.

Each CIF data file should contain a reference to one or more dictionary files against which the file may be validated. At the very least this will be **_audit_conform_dict_name** (**_audit_conform.dict_name** for DDL2 files) (*N*). ***_version** (*V*) and ***_location** (*L*) are optional. In the event that no dictionaries are specified, the default validation dictionary should be that identified as having *N* = `cif_core.dic` and *V* = '.' (*i.e.* the most recent version of the core dictionary). Since dictionaries are intended always to be extended, it is normally enough just to specify the name (and possibly the location).

This default is appropriate for most well formed CIFs, but if it is important to provide formal validation of old CIFs conforming to the earliest printed specification, which used the now-deprecated units extension convention, the dictionary cif_compat.dic may also be added to the default list (Section 3.1.5.4.3).

There is a difficulty associated with assuming this default for CIFs containing DDL2 data names. At present, the DDL2 version of the core dictionary does not exist as a separate file. Most existing CIFs built on the DDL2 model conform to the macromolecular (mmCIF) dictionary, and so best current working practice is to assume a default validation dictionary for DDL2-style CIFs with *N* = `mmcif_std.dic` and *V* = '.' (*i.e.* the most recent version of the mmCIF dictionary), since this includes the core data names as a subset. However, to anticipate future developments, it is suggested that applications built to validate DDL2 files first search the register for a default entry with *N* = `cif_core.dic`, *V* = '.' and a value of 2 or higher for the relevant DDL version:

```
loop_
  _cifdic_dictionary.name
  _cifdic_dictionary.version
  _cifdic_dictionary.DDL_compliance
  cif_core.dic      .     2.1.2
```

A software application validating against CIF dictionaries should attempt to locate and validate against the referenced dictionaries in the order cited in the data file, according to the following procedure. The terms 'warning' and 'error' in this procedure are not necessarily messages to be delivered to a user. They may be handled as condition codes or return values delivered to calling procedures instead.

If *N*, *V* and *L* are all given, try to load the file from the location *L*, or a locally cached copy of the referenced file. If this fails, raise a warning. Then search the dictionary register for entries matching the given *N* and *V*. (An appropriate strategy would be to search a locally cached copy of the register, and to refresh that local copy with the latest version from the network if the search fails.) If a successful match is made, try to retrieve the file from the location given by the matching entry in the register (or a locally cached copy with the same *N* and *V* previously fetched from the location specified in the register). If this fails, try to load files identified from the register with the same *N* but progressively older versions *V* (version numbering takes the form *n.m.l* . . ., where *n*, *m*, *l*, . . . are integers referring to progressively less significant revision levels). Version '.' (meaning the current version) should be accessed before any other numbered version. If this fails, raise a warning indicating that the specified dictionary could not be located.

If *N* and *V* but not *L* are given, try to load locally cached or master copies of the matching dictionary files from the location specified in the register file, in the order stated above, *viz*: (i) the version number *V* specified; (ii) the version with version number indicated as '.'; (iii) progressively older versions. Success in other than the first instance should be accompanied by a warning and an indication of the revision actually loaded.

If only *N* is given, try to load files identified in the register by (i) the version with version number indicated as '.'; (ii) progressively older versions.

If all efforts to load a referenced dictionary fail, the validation application should raise a warning.

If all efforts to load all referenced dictionaries fail, the validation application should raise an error.

For any dictionary file successfully loaded according to this protocol, the validation application must perform a consistency check by scanning the file for internal identifiers (`_dictionary_name`, `_dictionary_version` or the DDL2 equivalents) and ensuring that they match the values of *N* and *V* (where *V* is not '.'). Failure in matching should raise an error.

3.1.9. Composite dictionaries

The dictionaries referenced by a data file are those that contain the definitions of the data names used in the data file. Typically these include or consist entirely of public dictionaries that are necessarily permissive in the range of values allowed for data items. However, the power and flexibility of validating against machine-readable dictionaries could be harnessed by applications that need to impose stricter validation criteria. For example, the core dictionary permits an enumeration range of 0 to 8 for `_atom_site_attached_hydrogens`, but one might wish to validate a data set describing well behaved organic molecules where anything above 4 is almost certainly an error. It would be helpful to have a validation dictionary identical to the core dictionary except for this enumeration range; however it would be inefficient to create an alternative dictionary of the same size simply for this one change. In Section 3.1.9.1, we consider how to build a dictionary file that includes the bulk of the content of the public dictionaries cited in the CIF, together with modifications in local dictionary files to allow alternative specifications of what constitutes a 'valid' data item.

Proper applications of this approach include restricting the enumeration range specified for an item in a public dictionary; enforcing a more strict data typing than allowed by the parent dictionary; storing a list of all data names (including local ones) permitted in a CIF; or adding to existing dictionary entries references to local data items in an extension dictionary. An example of the latter application would be the addition of a `_list_link_child` entry to a public definition to accompany the introduction of a new child category in a local dictionary. The protocol to be described does not prohibit other applications, but care must be taken to generate dictionaries that retain internal consistency and are properly parsable by standard validation tools.

3.1.9.1. A dictionary merging protocol

The following protocol describes the construction of a *composite*, or *virtual*, dictionary by merging and overlaying fragments of a local validation dictionary and the public dictionaries referenced from within a data file. The term 'dictionary fragment' refers here to a physical disk file which contains one or more data blocks or save frames (according to whether the relevant data model is DDL1 or DDL2) containing complete or partial sets of attributes associated with data names identified in the relevant dictionary data block or save frame through the item `_name` (DDL1) or `_item.name` (DDL2).

(i) Assemble and load all dictionary fragments against which the current data block will be validated. The order of presentation is important. Complete dictionaries referenced by a data file should be assembled in the order cited. A dictionary validation application may then accept a list of additional dictionary fragments to PREPEND to, REPLACE or APPEND to each file in the list of cited dictionaries. In most applications, it will be appropriate to append to or replace attributes defined in a public dictionary, and the PREPEND operation is presented only for completeness.

(ii) Define three modes in which conflicting data names in the aggregate dictionary file may be resolved, called STRICT, REPLACE and OVERLAY.

(iii) Scan the aggregate dictionary fragments in the order of loading. Assemble for each defined data name a composite definition frame (data block or save frame as appropriate) as follows, depending on the mode in which the validation application is operating:

STRICT: If a data name appears to be multiply defined, generate a fatal error. This mode permits the interpolation of local dictionaries that do not attempt to modify the attributes of public data items.

REPLACE: All attributes previously stored for the conflicting data name are deleted, and only the attributes in the later data block (or save frame) containing the definition are preserved. This mode permits the complete redefinition of public data names and is not appropriate for validation of CIFs to be archived. Its main use would be in testing modifications of individual definition frames outside the parent dictionary.

OVERLAY: New attributes are added to those already stored for the data name; conflicting attributes replace those already stored. This is the standard mechanism for modifying attributes for application-specific validation purposes.

This protocol allows the creation of a coherent virtual dictionary from several different dictionary files or fragments. Although it must be used with care, it permits different levels of validation based on dictionary-driven methods without modifying the original dictionary files themselves.

Example 3.1.9.1. *A standard CIF dictionary definition block.*

```
data_atom_site_attached_hydrogens
    _name              '_atom_site_attached_hydrogens'
    _category            atom_site
    _type                numb
    _list                yes
    _list_reference      '_atom_site_label'
    _enumeration_range   0:8
    _enumeration_default 0
    loop_ _example
          _example_detail  2    'water oxygen'
                           1    'hydroxyl oxygen'
                           4    'ammonium nitrogen'
    _definition
;       The number of hydrogen atoms attached
        to the atom at this site excluding any
        hydrogen atoms for which coordinates
        (measured or calculated) are given.
;
```

Example 3.1.9.2. *A modified data attribute for overlaying a public definition.*

```
data_atom_site_attached_hydrogens_restricted
    _name              '_atom_site_attached_hydrogens'
    _enumeration_range   0:4
```

As an example, consider the core CIF dictionary definition mentioned above of the number of hydrogen atoms that might be attached to an atom site (Example 3.1.9.1).

For a particular application, any structures reporting more than four attached hydrogen atoms might be considered as invalid. A validation program to satisfy this requirement might therefore build a composite dictionary from the public cif_core.dic, which contains the definition in Example 3.1.9.1, and the fragment of Example 3.1.9.2, processed in APPEND/OVERLAY modes.

3.1.9.2. Protocol implementation

At the time of publication (2005), there is no reference implementation for this protocol, and so the proper treatment of the fine details of merging and overlay operations is not available. The following guidelines outline the first steps in an implementation under DDL1.4.

The description assumes that a composite dictionary is to be assembled from two public dictionaries, a.dic and b.dic, and a local dictionary mod.dic that includes some modifications to the definitions in one or both of the public dictionaries (and is therefore processed in OVERLAY mode). It is assumed that the composite dictionary will be written to disk as a separate file, virtual.dic, although in practice applications may simply construct the image of the composite dictionary in memory.

(1) Each contributing dictionary fragment should have at most one data block containing the data names `_dictionary_name` and `_dictionary_version` (with, optionally, `_dictionary_update` and `_dictionary_history`). The `*_name` and `*_version` together identify the dictionary file uniquely and should match the corresponding entries in the IUCr register if this is a public dictionary. This information is conventionally stored in a data block named `data_on_this_dictionary`.

In DDL1.4, all four of the items `_dictionary_name`, `*_version`, `*_update` and `*_history` are scalars, *i.e.* may not be looped. Hence a new dictionary identifier section in virtual.dic may be constructed as follows.

(i) Create a data block `data_on_this_dictionary` at the beginning of virtual.dic.

(ii) If a name for the composite dictionary is supplied (*via* a command-line switch, for example), write this as the value of `_dictionary_name`; otherwise generate a pseudo-unique string (*e.g.* concatenate the computer identifier string, process number and current date string).

(iii) If a dictionary version number is supplied (*via* a command-line switch, for example), write this as the value of `_dictionary_version`; otherwise supply the value '1.0'.

(iv) Supply the current date in the format *yyyy-mm-dd* as the value of `_dictionary_update`.

(v) Create a composite `_dictionary_history` by concatenation of the individual `_dictionary_history` fragments. The application may add details of the current merge operation to the history field.

(2) There is no significance to the ordering of data blocks containing definitions in dictionaries, although they are conventionally sorted alphabetically. For convenience, data blocks should be written out in the order in which they are encountered in the input primitive dictionary files, except that definitions modified by subsequent entries remain in their initial location.

(3) In STRICT mode, if the same value of `_name` is present in two or more data blocks, the composite dictionary is invalid and the application should raise a fatal error. Otherwise the composite dictionary simply contains the aggregate definitions from multiple input dictionaries.

(4) In REPLACE mode, a stored definition block is discarded and replaced by a new definition of the item referenced by `_name`.

(5) For the OVERLAY mode (assumed in the present discussion), the following procedure is proposed. Load a data block from the first dictionary file. Locate the `_name` tag. (Because `_name` may be looped, a data block may contain definitions for more than one data name. For convenience, we consider only the case of a data block containing a single value of `_name`. In any event, it is possible to separate a set of looped definitions into individual data blocks, each defining only one of the data names in the initial `_name` loop.) Search the next dictionary file for a data block containing the same value of `_name`. Load the contents of that data block.

(i) If the new data block contains only data items that do not appear in the first data block, they are simply concatenated with those already present.

(ii) If the new data block contains a scalar data item already present in the first data block (*i.e.* with `_list no`), discard the stored attributes.

(iii) If the new data block contains data items that may be looped and that occur in the first data block, build a new composite table of values in the following way: (*a*) construct a valid loop header if necessary; (*b*) do not repeat identical sets of values (*i.e.* collapse identical table rows); (*c*) if it is possible to identify the category key, then raise a fatal error if there are identical instances of a key value [after the normalization of step (*b*) has occurred]; (*d*) else append new rows to the table.

When the new composite data block has been built according to these principles, search the next dictionary file specified and repeat.

3.1.10. Public CIF dictionaries

So far, seven CIF dictionaries have been published by the IUCr with COMCIFS approval. They are described in the remaining chapters in this part of the volume. This section provides an overview of the large-scale structure of these dictionaries and forms a general introduction to Chapters 3.2 to 3.8.

The public CIF dictionaries have been constructed by experts in a number of different crystallographic fields. They are intended to serve the individual fields in which they have been commissioned and therefore vary in character depending on the requirements

Table 3.1.10.1. *High-level grouping of categories by dictionary*

Category groups are organized into families by common function and purpose.

cif_core.dic	cif_pd.dic	cif_ms.dic	cif_rho.dic	mmcif_std.dic	cif_img.dic	cif_sym.dic
(*a*) Experimental measurements						
					ARRAY	
					AXIS	
CELL		CELL		CELL		
DIFFRN		DIFFRN		DIFFRN	DIFFRN	
EXPTL		EXPTL		EXPTL		
	PD_CALIB					
	PD_CHAR					
	PD_DATA					
	PD_INSTR					
	PD_MEAS					
	PD_PREP					
	PD_SPEC					
(*b*) Analysis						
	PD_CALC					
	PD_PEAK					
	PD_PROC					
				PHASING		
REFINE		REFINE		REFINE		
REFLN	REFLN	REFLN		REFLN		
(*c*) Structure						
ATOM		ATOM	ATOM	ATOM		
CHEMICAL				CHEMICAL		
				CHEM_COMP		
				CHEM_LINK		
				ENTITY		
GEOM		GEOM		GEOM		
	PD_PHASE					
				STRUCT		
SYMMETRY		SYMMETRY		SYMMETRY		SPACE_GROUP
VALENCE				VALENCE		
(*d*) Publication						
CITATION				CITATION		
COMPUTING				COMPUTING		
DATABASE				DATABASE		
JOURNAL				JOURNAL		
PUBL				PUBL		
				SOFTWARE		
(*e*) File metadata						
AUDIT		AUDIT		AUDIT		
	PD_BLOCK					

and practices of each field. Here we provide a general framework within which the category groups of each separate dictionary may be described.

3.1.10.1. Categories and category groups

The only formal unit of classification common to all CIF dictionaries is the *category*. For example, in the core CIF dictionary information about the chemical and physical properties of the different atomic species in a crystal cell is collected in a few data names such as **_atom_type_oxidation_number** which belong to the same category, in this case the ATOM_TYPE category. As described in Section 3.1.5.3, it is conventional (although not mandatory) that CIF data names begin with components corresponding to the name of the category to which they belong.

The term *category* as used in CIF dictionaries has a technical meaning which constrains its normal use in grouping items that are understood to have a 'natural' relationship. In a CIF, only items belonging to the same category may appear together in the same looped list. This means, for example, that data items describing collective properties of the atom sites in the lattice (such as the number of atoms of each atomic species in the unit cell) must be assigned to a different category from the data items that describe the properties of the individual sites. Hence the properties of individual sites (such as the positional coordinates defined by **_atom_site_fract_x** *etc.*) belong to the ATOM_SITE category, while the transformation matrix between Cartesian and fractional components (expressed by a collection of data names such as **_atom_sites_fract_tran_matrix_11**) belong to the ATOM_SITES category. Clearly, the category names have been chosen to be similar to reflect their close relationship, while the EXPTL category containing data names such as **_exptl_crystal_colour** is named quite differently. It is natural to wish to describe related categories in a common higher level of classification, and indeed *category groups* exist as formal components of DDL2-structured dictionaries. We shall, however, refer informally to 'category groups' in discussions of DDL1 dictionaries as collections of categories with a close relationship that is usually implicit in their names.

3.1.10.2. Overview of category classification

Table 3.1.10.1 provides an informal classification at a high level of the category groups represented in each of the CIF dictionaries in this volume. Related category groups are clustered within the table in families sharing a common function. The five families (*a*) to (*e*) in Table 3.1.10.1 refer to: the crystallographic experiment itself; the processing and analysis of data from the experiment; the derived structural model; the reporting and publication of the results; and general auditing of the file itself, its purpose, authorship, history and links to other data sets, *i.e.* the file metadata. Detailed discussions of the individual categories (and formal category groups for DDL2 dictionaries) will be found in the relevant chapters in the rest of this part of the volume.

Table 3.1.10.1 shows the different characters of the seven dictionaries. The macromolecular dictionary (mmcif_std.dic; Chapter 4.5) contains an embedded version of the core dictionary

(cif_core.dic; Chapter 4.1) in DDL2 format and so includes all the categories defined in the core. However, it extends the description of the structural model extensively by introducing families of categories for the description of chemical components of a macromolecular structure (ENTITY) and for the detailed description of the structure itself (STRUCT). New categories are also introduced to describe the phasing of the structure and the SOFTWARE category allows the inclusion of more details of computational techniques than the core COMPUTING category does.

The other dictionaries are purely extensions which either introduce new data names (and occasionally new categories) into existing category groups or, where necessary, introduce completely new groups of categories.

The powder dictionary (cif_pd.dic; Chapter 4.2) contains several new category groups reflecting the need for substantially different methods of describing the experiment and analysing the data, as well as a need for the structural model to be able to handle multiple crystalline phases. The modulated structures dictionary (cif_ms.dic; Chapter 4.3) introduces no new category groups, but does introduce several new data names and categories within the existing framework. The electron density dictionary (cif_rho.dic; Chapter 4.4) introduces two new categories within an existing category group. The image CIF dictionary (cif_img.dic; Chapter 4.6) has several new categories that characterize arrays of data from two-dimensional X-ray detectors and the consequent detailed descriptions of the relevant axes within the experimental setup. The symmetry dictionary (cif_sym.dic; Chapter 4.7) was commissioned specifically to replace the symmetry categories in the core dictionary with a more detailed treatment.

References

Bray, T., Paoli, J. & Sperberg-McQueen, C. M. (1998). *Extensible Markup Language (XML) 1.0*. W3C Recommendation 10-February-1998. http://www.w3.org/TR/1998/REC-xml-19980210.

CrossRef (2004). *Query spec.* http://www.crossref.org/03libraries/25query_spec.html.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R., Spadaccini, N., Castleden, I. R., du Boulay, D. & Westbrook, J. D. (2002). *StarDDL: towards the unification of Star dictionaries. Acta Cryst.* A**58** (Suppl.), C256.

W3C (2004). *Extensible Markup Language (XML)*. http://www.w3c.org/XML/.

3.2. Classification and use of core data

BY S. R. HALL, P. M. D. FITZGERALD AND B. MCMAHON

3.2.1. Introduction

This chapter is concerned with the classification and organization of data items defined in the core CIF dictionary (Chapter 4.1). The core dictionary, as its name suggests, is central to the definition of data items found in most CIFs. It defines the measured and derived items common to most crystallographic experiments, analyses and publications, and, in particular, those items characterizing a classical single-crystal X-ray diffraction determination of a small-molecule or inorganic structure. As the nature of crystallographic studies evolves, so do the data items needed to describe them. New data names are introduced as needed to describe new techniques or technologies or simply to provide more details of subjects already covered. In addition, the developers of specialist dictionaries may find that some of the items they define have a wider application and propose that these items be added to the core dictionary instead.

Core data items are defined with two formalisms. The core dictionary, as presented in Chapter 4.1, defines core data items exclusively using the data definition language DDL1 (described in Chapter 2.5). However, core data items are also embedded within the macromolecular CIF dictionary presented in Chapter 4.5 using the data definition language DDL2 (described in Chapter 2.6). Because the revision cycles of the core and mmCIF dictionaries are not synchronized, at any one time the mmCIF dictionary may not include the complete set of data items in the current core dictionary. The mmCIF dictionary described in this volume includes the full content of core CIF dictionary version 2.3.1, also described in this volume.

The discussion in this chapter will concentrate on the current DDL1 version of the core dictionary (version 2.3, released on 4 October 2003 and reissued with minor amendments as version 2.3.1 in this volume). There will be some discussion of the more formal approach to the classification of data items that DDL2 permits.

In accordance with the scheme given in Table 3.1.10.1, groups of categories of data items in the core dictionary will be classified under the headings *Experimental measurements* (Section 3.2.2), *Analysis* (Section 3.2.3), *Atomicity, chemistry and structure* (Section 3.2.4), *Publication* (Section 3.2.5) and *File metadata* (Section 3.2.6). To help the reader relate the thematic order of the discussion of these categories to the alphabetic layout of the dictionary, the category structure of the core dictionary is summarized in Table 3.2.1.1 and is listed in full in Appendix 3.2.1. The appendix also lists for each category the section of this chapter in which the category is described.

The data items contained within each category are listed in the detailed commentary below. Where relevant, the data item or items that represent a unique identifier for a looped list ('category keys') are listed first and are marked by a bullet (•). Note that category keys are defined more formally in the mmCIF dictionary (see Chapter 2.6 and the discussion of categories in Section 3.1.6.4). The remaining data items in each category are listed alphabetically.

Table 3.2.1.1. *Category groups defined in the core CIF dictionary*

The groups are listed in the order in which they are described in this chapter.

Section	Category group	Subject covered
(*a*) Experimental measurements		
3.2.2.1	CELL	Unit cell
3.2.2.2	DIFFRN	Diffraction experiment
3.2.2.3	EXPTL	Experimental conditions
(*b*) Analysis		
3.2.3.1	REFINE	Refinement procedures
3.2.3.2	REFLN	Reflection measurements
(*c*) Atomicity, chemistry and structure		
3.2.4.1	ATOM	Atom sites
3.2.4.2	CHEMICAL	Chemical properties and nomenclature
3.2.4.3	GEOM	Geometry of atom sites
3.2.4.4	SYMMETRY	Symmetry information
3.2.4.5	VALENCE	Bond-valence information
(*d*) Publication		
3.2.5.1	CITATION	Bibliographic references
3.2.5.2	COMPUTING	Computational details of the experiment
3.2.5.3	DATABASE	Database information
3.2.5.4	JOURNAL	Journal housekeeping
3.2.5.5	PUBL	Contents of a published article
(*e*) File metadata		
3.2.6	AUDIT	Dictionary maintenance and identification

3.2.2. Experimental measurements

Crystallographic archive files predating CIF were often constructed to serve the purposes of a particular software program or suite and stored the data generated by an experiment without providing a full record of the conditions under which the data were obtained. This is not unique to crystallography: many data formats make no provision for the metadata – information about the procedures for gathering and analysing data – that give context and in many cases significance to the numeric values. A specific goal of the design of CIF was to treat such supporting information as essential elements of the whole collection of information relating to a structure determination, rather than as optional and poorly defined metadata. There are therefore many categories in the core dictionary that relate to experimental conditions and apparatus, and these categories are discussed in this section. They include the categories in the DIFFRN group describing the traditional crystallographic diffraction experiment (typically a single-crystal laboratory-based X-ray determination, but increasingly including synchrotron experiments and experiments using other radiation types). There are also categories that describe and characterize the crystal used in the experiment and those that characterize the unit cell, since the experimental determination of the cell parameters is an essential part of the full structure-determination experiment.

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, 6009, Australia; PAULA M. D. FITZGERALD, Merck Research Laboratories, Rahway, New Jersey, USA; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

3.2.2.1. Crystal cell parameters and measurement conditions

The categories describing the crystal unit cell and its determination are as follows:

CELL group
- CELL
- CELL_MEASUREMENT_REFLN

The data items in these categories are as follows:

(*a*) CELL
- _cell_angle_alpha
- _cell_angle_beta
- _cell_angle_gamma
- _cell_formula_units_Z
- _cell_length_a
- _cell_length_b
- _cell_length_c
- _cell_measurement_pressure
- _cell_measurement_radiation
- _cell_measurement_reflns_used
- _cell_measurement_temperature
- _cell_measurement_theta_max
- _cell_measurement_theta_min
- _cell_measurement_wavelength
- _cell_reciprocal_angle_alpha
- _cell_reciprocal_angle_beta
- _cell_reciprocal_angle_gamma
- _cell_reciprocal_length_a
- _cell_reciprocal_length_b
- _cell_reciprocal_length_c
- _cell_special_details
- _cell_volume

(*b*) CELL_MEASUREMENT_REFLN
- • _cell_measurement_refln_index_h
- • _cell_measurement_refln_index_k
- • _cell_measurement_refln_index_l
- _cell_measurement_refln_theta

The bullet (•) *indicates a category key.*

The CELL category includes two groups of data names: those characterizing a crystal unit cell, and those describing the experimental conditions relating to the unit-cell determination. It is a feature of the formal definition of the *category* classification unit in CIF dictionaries that these may be classed within the same category, whereas the Miller indices of the reflections used in the measurement of the unit cell belong to a different category. An argument could be made for dividing the CELL category into two categories to reflect the division drawn above between the cell parameters and their determination. However, the CIF dictionaries have been designed to have as few separate categories as possible, subject to the constraint that data items that are looped together in the same list must belong to the same category.

The individual dictionary definitions of the data items in this category are unambiguous, with the possible exception of _cell_formula_units_Z, which records the number of complete *chemical* formula units present in the unit cell, and *not* the number of repetitions of the asymmetric unit. In some instances the value of Z could be less than the number of repetitions of the asymmetric unit, such as when an internally symmetric molecular unit is positioned on a symmetry element and spans multiple asymmetric units. Of course, Z can be greater than the number of repetitions of the asymmetric unit (*i.e.* $Z' > 1$).

Note that the value associated with the data item _cell_volume is not independent, but can be derived from the other cell parameters. Within the core dictionary there are many cases of derivable items, both because they have traditionally been reported separately and because the presence of redundant information allows cross checking of the internal consistency of the data set.

Data items in the CELL_MEASUREMENT_REFLN category record details about the reflections used to determine the crystallographic cell parameters. The key items in this list are marked with bullets; all three of the h, k, l values are needed to identify a reflection.

3.2.2.2. Data collection

The categories describing data collection are as follows:

DIFFRN group

General description (§3.2.2.2.1)
- DIFFRN

Apparatus and instrumentation before the crystal (§3.2.2.2.2)
- DIFFRN_ATTENUATOR
- DIFFRN_RADIATION
- DIFFRN_RADIATION_WAVELENGTH
- DIFFRN_SOURCE

Apparatus and instrumentation at the crystal (§3.2.2.2.3)
- DIFFRN_MEASUREMENT
- DIFFRN_ORIENT_MATRIX
- DIFFRN_ORIENT_REFLN

Apparatus and instrumentation after the crystal (§3.2.2.2.4)
- DIFFRN_DETECTOR

Intensity measurements (§3.2.2.2.5)
- DIFFRN_REFLN
- DIFFRN_REFLNS
- DIFFRN_REFLNS_CLASS
- DIFFRN_SCALE_GROUP
- DIFFRN_STANDARD_REFLN
- DIFFRN_STANDARDS

The category group related to the diffraction experiment is broad and includes details of the apparatus as well as the measurements. The individual categories are grouped together according to location within the experimental setup (see Fig. 3.2.2.1) or the measurement of intensities.

3.2.2.2.1. *General description*

The data items in this category are as follows:

DIFFRN
- _diffrn_ambient_environment
- _diffrn_ambient_pressure
- _diffrn_ambient_pressure_gt
- _diffrn_ambient_pressure_lt
- _diffrn_ambient_temperature
- _diffrn_ambient_temperature_gt
- _diffrn_ambient_temperature_lt
- _diffrn_crystal_treatment
- _diffrn_measured_fraction_theta_full
- _diffrn_measured_fraction_theta_max
- _diffrn_special_details
- _diffrn_symmetry_description

These data items give an overview of the diffraction experiment. They are intended to be independent of the instrument, techniques or methodology of the experiment.

The items describing the ambient environmental conditions are reasonably self-explanatory. They are often absent from a CIF, because an author has not thought it necessary to provide

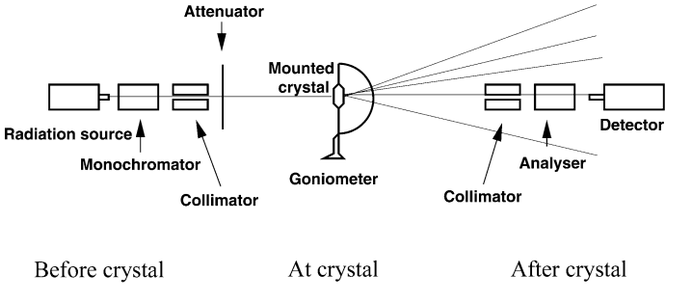


Fig. 3.2.2.1. Scheme of a diffraction experiment.

Example 3.2.2.1. *Attenuation of reflection intensities indicated by reference to attenuator scaling factors.*

```
loop_
     _diffrn_attenuator_code
     _diffrn_attenuator_material
     _diffrn_attenuator_scale
          1   Zr   16.976

loop_
     _diffrn_refln_index_h
     _diffrn_refln_index_k
     _diffrn_refln_index_l
     _diffrn_refln_attenuator_code
     _diffrn_refln_angle_chi
     _diffrn_refln_scan_rate
     _diffrn_refln_counts_bg_1
     _diffrn_refln_counts_total
     _diffrn_refln_counts_bg_2
     _diffrn_refln_scan_width
     _diffrn_refln_elapsed_time

0  0 -16  .   0.0 4.12  28 127  36 1.516 19.43
3  4  -4  1   0.0 1.03  69 459  73 1.516 2082.58
```

information for experiments conducted under 'normal' conditions of room temperature and pressure, and in a standard atmosphere. However, 'normal room temperature' may span a range of many degrees Kelvin and might have a non-negligible effect upon cell dimension measurements, so the temperature should be given. As there is significant variability in the ambient temperature at which laboratory experiments may be carried out, it is not appropriate to assign a default value for **_diffrn_ambient_temperature**, since any numeric value chosen as a default could be misconstrued as an experimentally determined value. If the ambient temperature has not been measured, an author may supply a best estimate of the ambient temperature with a suitable standard uncertainty. Alternatively, known upper and lower limits for the temperature may be given using **_diffrn_ambient_temperature_lt** and ***_gt**. The same considerations hold true for ambient pressure.

The default for **_diffrn_ambient_environment** may be understood as 'air', although formally it is impossible in the dictionary to specify a default for a free-text field.

The **_diffrn_measured_fraction_theta_*** items are provided in this category as an indication of the completeness of a set of reflection measurements. They are not as general as the other items in this category, as they apply only to monochromatic X-ray diffraction experiments, and they do not reflect the way macromolecular crystallographers tend to analyse the completeness of a data set as a function of resolution. When used, they must be accompanied by the value of the monochromatic radiation wavelength **_diffrn_radiation_wavelength** and relate to the maximum θ angle for which the measured reflection count is considered as complete (**_diffrn_reflns_theta_full**).

The other textual data items are provided for comment on other aspects of the handling of the crystal prior to the intensity measurement (**_diffrn_crystal_treatment**), observations on the diffraction point symmetry, systematic absences and inferred space group or superspace group relationships (**_diffrn_symmetry_description**) and any other comment on the intensity measurement process as a whole that cannot be accommodated elsewhere (**_diffrn_special_details**).

3.2.2.2.2. *Apparatus and instrumentation before the crystal*

The data items in these categories are as follows:

(*a*) DIFFRN_ATTENUATOR
```
• _diffrn_attenuator_code
  _diffrn_attenuator_material
  _diffrn_attenuator_scale
```

(*b*) DIFFRN_RADIATION
```
  _diffrn_radiation_collimation
  _diffrn_radiation_filter_edge
  _diffrn_radiation_inhomogeneity
  _diffrn_radiation_monochromator
  _diffrn_radiation_polarisn_norm
  _diffrn_radiation_polarisn_ratio
  _diffrn_radiation_probe
  _diffrn_radiation_type
  _diffrn_radiation_xray_symbol
```

(*c*) DIFFRN_RADIATION_WAVELENGTH
```
• _diffrn_radiation_wavelength_id
  _diffrn_radiation_wavelength
  _diffrn_radiation_wavelength_wt
```

(*d*) DIFFRN_SOURCE
```
† _diffrn_radiation_source
  _diffrn_source
  _diffrn_source_current
  _diffrn_source_details
  _diffrn_source_power
  _diffrn_source_size
  _diffrn_source_take-off_angle
  _diffrn_source_target
  _diffrn_source_type
  _diffrn_source_voltage
```

The bullet (•) *indicates a category key. The dagger* (†) *indicates a deprecated item, which should not be used in the creation of new CIFs.*

Attenuator properties are described by data items in the DIFFRN_ATTENUATOR category. Where an attenuator is used to reduce the intensity of an X-ray beam, this category may be used to describe the attenuator and its scaling factor. Details of multiple attenuator settings or materials can be included and each is identified by a code. A matching code value (**_diffrn_refln_attenuator_code**) appears in the list of intensities against each reflection that must be scaled by the appropriate attenuation factor. In Example 3.2.2.1, the intensity of the second reflection has been reduced using a zirconium attenuator and must be multiplied by 16.976 to place it on the same scale as the first (and other unattenuated intensities).

The DIFFRN_RADIATION category describes the radiation used in the diffraction experiment and its experimental handling by collimation and monochromatization before it interacts with the sample. [Post-sample treatment of the radiation beam after diffraction (including passage through any analyser or collimator) is described by data items in the complementary DIFFRN_DETECTOR category.] Many of the data items in this category are descriptive. Additional information about the generation of the radiation is also found in the DIFFRN_SOURCE category.

The use of **_diffrn_radiation_probe** is strongly recommended as an unambiguous indicator of the probing radiation or particle type (its permitted values are `x-ray`, `neutron`, `electron` and `gamma`). The similar-sounding data name **_diffrn_radiation_type** allows for a more detailed description of the radiation type, such as `white-beam` or (using the CIF code for the Greek character α, \a) `'Cu K\a'` for copper $K\alpha$ radiation. In the case of monochromatic (or near-monochromatic) X-radiation, a better representation is given by the use of **_diffrn_radiation_xray_symbol**, which can have one of a limited number of values expressing the X-ray wavelength according to IUPAC conventions (*e.g.* K-L_3, corresponding to the older Siegbahn notation $K\alpha_1$). If this data item is used, the element used as the X-ray generator target must also be specified using the data item **_diffrn_source_target**. Software for reading CIFs should be aware of these two alternative representations.

If the radiation beam is monochromatic, the wavelength can be provided using **_diffrn_radiation_wavelength**.

For a polychromatic beam, the other data items in the DIFFRN_RADIATION_WAVELENGTH category allow different wavelength components and an associated weighting factor for each component to be listed. In the list of experimental intensity measurements from a polychromatic beam (the DIFFRN_REFLN category, discussed below), each reflection has an associated **_diffrn_refln_wavelength_id** that must match the corresponding **_diffrn_radiation_wavelength_id** in this list.

The DIFFRN_SOURCE category specifies the characteristics of the radiation source in the experiment and is closely related to the DIFFRN_RADIATION category, which is concerned with the handling of the radiation beam before it reaches the specimen. (The now-deprecated data name **_diffrn_radiation_source** shows that there was no formal separation of the descriptions of the radiation generator and the radiation in the first release of the core dictionary.)

The general class of radiation is specified by the data name **_diffrn_source**, which is a free-text field. Typical entries would be 'sealed X-ray tube', 'nuclear reactor', 'synchrotron', 'spallation source', 'rotating-anode X-ray tube' or 'electron microscope'. It is clear that the category could describe non-X-ray experiments, but several of the data names within the category (*e.g.* **_diffrn_source_target**) have meanings that are specific to an X-ray experiment. New data names might be introduced if experiments using other radiation types become more common. For now, details that a user wishes to record that are not properly described by the existing data names may be stored in the **_diffrn_source_details** field.

3.2.2.2.3. *Apparatus and instrumentation at the crystal*

The data items in these categories are as follows:

(*a*) DIFFRN_MEASUREMENT

_diffrn_measurement_details
_diffrn_measurement_device
_diffrn_measurement_device_details
_diffrn_measurement_device_type
_diffrn_measurement_method
_diffrn_measurement_specimen_support

(*b*) DIFFRN_ORIENT_MATRIX

_diffrn_orient_matrix_type
_diffrn_orient_matrix_UB_11
_diffrn_orient_matrix_UB_12
_diffrn_orient_matrix_UB_13
_diffrn_orient_matrix_UB_21
_diffrn_orient_matrix_UB_22
_diffrn_orient_matrix_UB_23
_diffrn_orient_matrix_UB_31
_diffrn_orient_matrix_UB_32
_diffrn_orient_matrix_UB_33

(*c*) DIFFRN_ORIENT_REFLN

• **_diffrn_orient_refln_index_h**
• **_diffrn_orient_refln_index_k**
• **_diffrn_orient_refln_index_l**
_diffrn_orient_refln_angle_chi
_diffrn_orient_refln_angle_kappa
_diffrn_orient_refln_angle_omega
_diffrn_orient_refln_angle_phi
_diffrn_orient_refln_angle_psi
_diffrn_orient_refln_angle_theta

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key.*

The DIFFRN_MEASUREMENT category currently concerns specifically the mounting of the crystal and the details of the goniometer or other device on which it is mounted, with the exception of **_diffrn_measurement_method**, which is defined simply as the 'method used to measure intensities'. In practice, for a typical single-crystal diffractometer setup this field is generally used to specify the scan type, as in Example 3.2.2.2, where the CIF code for the Greek character θ, \q, is used to indicate $\theta/2\theta$ scans.

Example 3.2.2.2. *An indication of the scan type of a diffractometer-based experiment.*

```
_diffrn_measurement_method
    'profile data from \q/2\q scans'
```

The orientation matrix gives the transformation between coordinates in a crystal-centric reference frame and those referred to the diffractometer axes. The data items defined in the DIFFRN_ORIENT_MATRIX category can be used to store the values in the matrix as recorded on an individual diffractometer and a reference to the convention used (in **_diffrn_orient_matrix_type**). However, the reference is not by itself sufficient to understand the transformation without additional external knowledge of the convention. Authors are encouraged to provide a full description of the convention in the text field **_diffrn_orient_matrix_type**.

The terminology UB refers to the conventional designation of the matrix relating reciprocal space and the reference frame of a diffractometer, calculated as the product of the orientation matrix **U** and the material matrix **B** by the method of Busing & Levy (1967).

The reflections used to determine the orientation matrix can be listed in the category DIFFRN_ORIENT_REFLN. As discussed above, this list is useful for analysing the results on a diffractometer of known type, but is not useful if the convention for establishing the individual terms of the orientation matrix is not known.

3.2.2.2.4. *Apparatus and instrumentation after the crystal*

The data items in this category are as follows:

DIFFRN_DETECTOR

_diffrn_detector
_diffrn_detector_area_resol_mean
_diffrn_detector_details
_diffrn_detector_dtime
_diffrn_detector_type
† **_diffrn_radiation_detector**
† **_diffrn_radiation_detector_dtime**

The dagger (†) *indicates a deprecated item, which should not be used in the creation of new CIFs.*

The DIFFRN_DETECTOR category is intended to describe the detector used to measure the scattered radiation, including any analyser and post-sample collimation. There are not many data names in this category, as it is not often necessary to know a lot about the detector beyond its make, model or name if it is made by a well known manufacturer. A record of the detector deadtime (**_diffrn_detector_dtime**) and the resolution of an area detector (**_diffrn_detector_area_resol_mean**) are useful details worth recording explicitly; other unusual or noteworthy details may be recorded in **_diffrn_detector_details**.

The deprecated items (retained for compatibility with the original release version) have been replaced by **_diffrn_detector** and **_diffrn_detector_dtime** to produce names better matched to the formal category assignment.

3.2.2.2.5. *Intensity measurements*

The data items in these categories are as follows:

(*a*) DIFFRN_REFLN

• **_diffrn_refln_index_h**
• **_diffrn_refln_index_k**
• **_diffrn_refln_index_l**

```
  _diffrn_refln_angle_chi
  _diffrn_refln_angle_kappa
  _diffrn_refln_angle_omega
  _diffrn_refln_angle_phi
  _diffrn_refln_angle_psi
  _diffrn_refln_angle_theta
  _diffrn_refln_attenuator_code
        → _diffrn_attenuator_code
  _diffrn_refln_class_code
        → _diffrn_reflns_class_code
  _diffrn_refln_counts_bg_1
  _diffrn_refln_counts_bg_2
  _diffrn_refln_counts_net
  _diffrn_refln_counts_peak
  _diffrn_refln_counts_total
  _diffrn_refln_crystal_id
        → _exptl_crystal_id
  _diffrn_refln_detect_slit_horiz
  _diffrn_refln_detect_slit_vert
  _diffrn_refln_elapsed_time
  _diffrn_refln_intensity_net
† _diffrn_refln_intensity_sigma
  _diffrn_refln_intensity_u
  _diffrn_refln_scale_group_code
        → _diffrn_scale_group_code
  _diffrn_refln_scan_mode
  _diffrn_refln_scan_mode_backgd
  _diffrn_refln_scan_rate
  _diffrn_refln_scan_time_backgd
  _diffrn_refln_scan_width
  _diffrn_refln_sint/lambda
  _diffrn_refln_standard_code
        → _diffrn_standard_refln_code
  _diffrn_refln_wavelength
  _diffrn_refln_wavelength_id
        → _diffrn_radiation_wavelength_id
```

(*b*) DIFFRN_REFLNS

```
  _diffrn_reflns_av_R_equivalents
† _diffrn_reflns_av_sigmaI/netI
  _diffrn_reflns_av_unetI/netI
  _diffrn_reflns_limit_h_max
  _diffrn_reflns_limit_h_min
  _diffrn_reflns_limit_k_max
  _diffrn_reflns_limit_k_min
  _diffrn_reflns_limit_l_max
  _diffrn_reflns_limit_l_min
  _diffrn_reflns_number
  _diffrn_reflns_reduction_process
  _diffrn_reflns_resolution_full
  _diffrn_reflns_resolution_max
  _diffrn_reflns_theta_full
  _diffrn_reflns_theta_max
  _diffrn_reflns_theta_min
  _diffrn_reflns_transf_matrix_11
  _diffrn_reflns_transf_matrix_12
  _diffrn_reflns_transf_matrix_13
  _diffrn_reflns_transf_matrix_21
  _diffrn_reflns_transf_matrix_22
  _diffrn_reflns_transf_matrix_23
  _diffrn_reflns_transf_matrix_31
  _diffrn_reflns_transf_matrix_32
  _diffrn_reflns_transf_matrix_33
```

(*c*) DIFFRN_REFLNS_CLASS

```
• _diffrn_reflns_class_code
  _diffrn_reflns_class_av_R_eq
† _diffrn_reflns_class_av_sgI/I
  _diffrn_reflns_class_av_uI/I
  _diffrn_reflns_class_d_res_high
  _diffrn_reflns_class_d_res_low
  _diffrn_reflns_class_description
  _diffrn_reflns_class_number
```

(*d*) DIFFRN_SCALE_GROUP

```
• _diffrn_scale_group_code
  _diffrn_scale_group_I_net
```

(*e*) DIFFRN_STANDARD_REFLN

```
• _diffrn_standard_refln_index_h
• _diffrn_standard_refln_index_k
• _diffrn_standard_refln_index_l
  _diffrn_standard_refln_code
```

(*f*) DIFFRN_STANDARDS

```
  _diffrn_standards_decay_%
  _diffrn_standards_interval_count
  _diffrn_standards_interval_time
  _diffrn_standards_number
† _diffrn_standards_scale_sigma
  _diffrn_standards_scale_u
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. The dagger (†) indicates a deprecated item, which should not be used in the creation of new CIFs.

The DIFFRN_REFLN category describes the measured reflections in a diffraction experiment. Example 3.2.2.3 shows a listing from a CAD-4 single-crystal diffractometer.

Note that the data in this list refer to the raw measurements as acquired at the time of data collection. This is in contrast to the data in the REFLN list, which refer to the reflections after merging and scaling.

The meanings of most of the data names can be deduced by inspection of this example. Full definitions are given in the dictionary.

However, the category also contains a number of data items which are used to group blocks of reflections with additional properties described by data items in other categories. For example, a number of reflections in the list might share a common value of `_diffrn_refln_scale_group_code`; this value would link to a description in the DIFFRN_SCALE_GROUP category of the scaling factor that needs to be applied to this group of reflections to bring all intensities in the list on to a common scale. (For example, intensities might be obtained from individual films in a multi-film data set or from a number of separate crystals.)

Likewise, individual reflections might be marked to indicate that they were monitored as standards during the course of the

Example 3.2.2.3. *Listing of experimental diffraction intensities.*

```
loop_
     _diffrn_refln_index_h
     _diffrn_refln_index_k
     _diffrn_refln_index_l
     _diffrn_refln_angle_chi
     _diffrn_refln_scan_rate
     _diffrn_refln_counts_bg_1
     _diffrn_refln_counts_total
     _diffrn_refln_counts_bg_2
     _diffrn_refln_angle_theta
     _diffrn_refln_angle_phi
     _diffrn_refln_angle_omega
     _diffrn_refln_angle_kappa
     _diffrn_refln_scan_width
     _diffrn_refln_elapsed_time
0  0 -16  0. 4.12  28 127  36  33.157  -75.846
          16.404   50.170  1.516 19.43
0  0 -15  0. 4.12  38 143  28  30.847  -75.846
          14.094   50.170  1.516 19.82
0  0 -14  0. 1.03 142 742 130  28.592  -75.846
          11.839   50.170  1.516 21.32
0  0 -13  0. 4.12  26 120  37  26.384  -75.846
           9.631   50.170  1.450 21.68
0  0 -12  0. 0.97 129 618 153  24.218  -75.846
           7.464   50.170  1.450 23.20
0  0 -11  0. 4.12  33 107  38  22.087  -75.846
           5.334   50.170  1.384 23.55
0  0 -10  0. 4.12  37 146  33  19.989  -75.846
           3.235   50.170  1.384 23.90
   # - - - - abbreviated - - - -
3  4  -4  0. 1.03  69 459  73  30.726  -53.744
          46.543  -47.552  1.516 2082.58
3  4  -5  0. 1.03  91 465  75  31.407  -54.811
          45.519  -42.705  1.516 2084.07
3 14  -6  0. 1.03  84 560  79  32.228  -55.841
          44.745  -38.092  1.516 2085.57
   # - - - - abbreviated - - - -
```

experiment, using the data name **_diffrn_refln_standard_code**. These standard reflections may be listed separately in the DIFFRN_STANDARD_REFLN category, in which case they are labelled by **_diffrn_standard_refln_code**, which must have values matching those assigned in the main list of intensities.

Apart from these specific classes of reflections, the intensity data may be binned according to different criteria (*e.g.* for modulated structures the intensities are often partitioned into classes with the same value of $m = \sum |m_i|$, where the m_i are the integer coefficients indexing diffraction vectors in an n-dimensional representation). The data name **_diffrn_refln_class_code** is provided as a link to the different classes of reflections defined in the DIFFRN_REFLNS_CLASS category.

The DIFFRN_REFLNS category describes collective properties of the set of experimental intensity measurements and follows the convention (common elsewhere in the dictionary) of having a name very similar to the related DIFFRN_REFLN category, but using a plural form of the relevant term in the composite name. While the individual DIFFRN_REFLN entries appear in a looped list, the items in the DIFFRN_REFLNS category are not looped.

This category describes properties of the *complete* measurement set; descriptions of specific portions of the complete set are handled by the DIFFRN_REFLNS_CLASS category.

Several of the items that appear in this category can be derived from the contents of the DIFFRN_REFLN lists, but it is often convenient to list them separately for ease of access and as a consistency check.

Note the definition of **_diffrn_reflns_number** as the total number of measured intensities *excluding* those classed as 'systematically absent' (reflections whose intensities are null as a consequence of crystallographic symmetry). There is no data item to specifically flag systematic absences (although one could assign a distinct **_diffrn_refln_class_code** value and define the relevant DIFFRN_REFLNS_CLASS). Because the measured diffraction data may (and often do) include reduced measurements and symmetry-equivalent reflection intensities, there is no formal way to check the value of **_diffrn_reflns_number** with dictionary-driven validation software. (Note that systematic absences *are* flagged in the structure-factor listing of the REFLN category.)

The data items in the DIFFRN_REFLNS_CLASS category record details about classes of reflections measured in the diffraction experiment. The user is free to assign classes according to arbitrary criteria; two specific cases, the marking of standard reflections and the clustering of intensities that need to be scaled by a common factor, have their own specific data items and associated categories, as discussed above. The example given in the dictionary (Example 3.2.2.4) describes a one-dimensional incommensurately modulated structure, where each reflection class is defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal lattice.

The DIFFRN_SCALE_GROUP category records scaling factors which must be applied to specific intensities in the DIFFRN_REFLN list to bring all the measurements on to a common scale (Example 3.2.2.5). The scale factor **_diffrn_scale_group_I_net** is the factor by which the relevant net values in the intensities list must be multiplied. The intensities to which it must be applied are those in the intensities list marked with a **_diffrn_refln_scale_group_code** that matches the corresponding **_diffrn_scale_group_code** in this category.

The DIFFRN_STANDARD_REFLN category allows a separate tabulation of the reflections used as standards. Note that the actual *measurements* on these reflections are stored alongside all the other measurements in the DIFFRN_REFLN list. The results of the analysis of the standard reflections are described by the DIFFRN_STANDARDS category.

Example 3.2.2.4. *Use of the DIFFRN_REFLNS_CLASS category to specify the main and satellite reflections collected for a modulated incommensurate structure*

```
loop_
     _diffrn_reflns_class_number
     _diffrn_reflns_class_d_res_high
     _diffrn_reflns_class_d_res_low
     _diffrn_reflns_class_av_R_eq
     _diffrn_reflns_class_code
     _diffrn_reflns_class_description
       1580 0.551 6.136 0.015 'Main'
          'm=0; main reflections'
       1045 0.551 6.136 0.010 'Sat1'
          'm=1; first-order satellites'
```

Example 3.2.2.5. *Scaling factors for reflections listed by group.*

```
loop_
     _diffrn_scale_group_code
     _diffrn_scale_group_I_net
      1    .86473
      2   1.0654
```

The DIFFRN_STANDARDS category describes the interval between measurements of the standard reflections and their overall intensity change (usually a decay, so that the relevant data name is **_diffrn_standards_decay_%**; this data item has a negative value if the final measured intensities are greater than the initial ones). The items assume a constant time interval (or number of counts) between the measurement of each standard and a single global value for the overall intensity change. If required, detailed tracking of the intensity change of individual standard reflections can be extracted from the DIFFRN_REFLN list provided the elapsed time at each measurement has been recorded (**_diffrn_refln_elapsed_time**).

3.2.2.3. Experimental measurements on the crystal

The categories describing experimental conditions are as follows:

EXPTL group
- EXPTL
- EXPTL_CRYSTAL
- EXPTL_CRYSTAL_FACE

The data items in these categories are as follows:

(*a*) EXPTL

_exptl_absorpt_coefficient_mu
_exptl_absorpt_correction_T_max
_exptl_absorpt_correction_T_min
_exptl_absorpt_correction_type
_exptl_absorpt_process_details
_exptl_crystals_number
_exptl_special_details

(*b*) EXPTL_CRYSTAL

- **_exptl_crystal_id**

_exptl_crystal_colour
_exptl_crystal_colour_lustre
_exptl_crystal_colour_modifier
_exptl_crystal_colour_primary
_exptl_crystal_density_diffrn
_exptl_crystal_density_meas
_exptl_crystal_density_meas_gt
_exptl_crystal_density_meas_lt
_exptl_crystal_density_meas_temp
_exptl_crystal_density_meas_temp_gt
_exptl_crystal_density_meas_temp_lt
_exptl_crystal_density_method

```
_exptl_crystal_description
_exptl_crystal_F_000
_exptl_crystal_preparation
_exptl_crystal_pressure_history
_exptl_crystal_recrystallization_method
_exptl_crystal_size_length
_exptl_crystal_size_max
_exptl_crystal_size_mid
_exptl_crystal_size_min
_exptl_crystal_size_rad
_exptl_crystal_thermal_history
```

(*c*) EXPTL_CRYSTAL_FACE

```
• _exptl_crystal_face_index_h
• _exptl_crystal_face_index_k
• _exptl_crystal_face_index_l
  _exptl_crystal_face_diffr_chi
  _exptl_crystal_face_diffr_kappa
  _exptl_crystal_face_diffr_phi
  _exptl_crystal_face_diffr_psi
  _exptl_crystal_face_perp_dist
```

The bullet (•) *indicates a category key. Where multiple items within a category are marked by a bullet, they must be taken together to form a compound key.*

The EXPTL category is rather broadly named, but in practice is used to record details about any absorption correction applied and, using **_exptl_special_details**, any other details of the experimental work prior to intensity measurement not specifically described by other data items (*e.g.* **_exptl_crystal_preparation**).

The data items in the EXPTL_CRYSTAL category are designed to record details of experimental measurements on the crystal or crystals used. Since it is usually the case that just one crystal is used throughout the experiment, the category is presented as if it comprises non-looped data names. However, details of a number of crystals may be looped together, in which case **_exptl_crystal_id** is used to identify the different crystals and acts as the category key.

When different crystals are used to collect diffraction intensities, it is likely that the intensities collected from each crystal would need to be scaled by different factors, as recorded by the DIFFRN_SCALE_GROUP category and the **_diffrn_refln_scale_group_code** used for each individual reflection. In these circumstances, it would be good practice to use matching values of **_diffrn_refln_scale_group_code** and **_exptl_crystal_id**, although this is not mandatory.

Note that the $F(000)$ value, which is often calculated as the integer number of electrons in the crystal unit cell, may contain dispersion contributions and is more properly calculated as

$$F(000) = \left[\left(\sum f_r\right)^2 + \left(\sum f_i\right)^2\right]^{1/2},$$

where f_r and f_i are, respectively, the real and imaginary parts of the scattering factors at $\theta = 0$ and the sum is taken over each atom in the unit cell.

The crystal colour may be given as free text using the data item **_exptl_crystal_colour**. Alternatively, the standardized names developed by the International Centre for Diffraction Data to classify specimen colours may be constructed from the items **_exptl_crystal_colour_lustre**, **_*_modifier** and **_*_primary**, each of which has a restricted set of specific values.

The EXPTL_CRYSTAL_FACE category records details of the crystal faces. The faces are defined by Miller indices and their perpendicular distances from the centre of rotation of the crystal may be recorded in millimetres. Absolute orientations with respect to the goniometer angle settings may also be recorded. The category is currently constructed in a way that cannot distinguish between multiple crystals.

3.2.3. Analysis

The categories relevant to the structural analysis are as follows:

Refinement techniques and results (§3.2.3.1)

REFINE group

- REFINE
- REFINE_LS_CLASS

The reflections used in the refinement (§3.2.3.2)

REFLN group

- REFLN
- REFLNS
- REFLNS_CLASS
- REFLNS_SCALE
- REFLNS_SHELL

In the small-molecule and inorganic studies for which the core dictionary was designed, phasing and structure solution are almost routine, and the dictionary provides few specific fields for recording the details of the structure solution process: **_atom_sites_solution_primary**, **_atom_sites_solution_secondary** and **_atom_sites_solution_hydrogens** (Section 3.2.4.1.2); **_computing_structure_solution** (Section 3.2.5.2); and **_publ_section_exptl_solution** (Section 3.2.5.5). (In contrast, the macromolecular CIF includes extensive details of phasing.) Refinement, however, still allows for a wide range of techniques, practices and interpretation, and there are a large number of data names to allow a full account of the refinement strategy to be given. To complement this, several categories exist to provide a detailed listing and annotation of the structure factors and their treatment according to shells of resolution or other sorting criteria.

3.2.3.1. Structure refinement

The data items in these categories are as follows:

(*a*) REFINE

```
  _refine_diff_density_max
  _refine_diff_density_min
  _refine_diff_density_rms
  _refine_ls_abs_structure_details
  _refine_ls_abs_structure_Flack
  _refine_ls_abs_structure_Rogers
  _refine_ls_d_res_high
  _refine_ls_d_res_low
  _refine_ls_extinction_coef
  _refine_ls_extinction_expression
  _refine_ls_extinction_method
  _refine_ls_goodness_of_fit_all
  _refine_ls_goodness_of_fit_gt
† _refine_ls_goodness_of_fit_obs
  _refine_ls_goodness_of_fit_ref
  _refine_ls_hydrogen_treatment
  _refine_ls_matrix_type
  _refine_ls_number_constraints
  _refine_ls_number_parameters
  _refine_ls_number_reflns
  _refine_ls_number_restraints
  _refine_ls_R_factor_all
  _refine_ls_R_factor_gt
† _refine_ls_R_factor_obs
  _refine_ls_R_Fsqd_factor
  _refine_ls_R_I_factor
  _refine_ls_restrained_S_all
  _refine_ls_restrained_S_gt
† _refine_ls_restrained_S_obs
† _refine_ls_shift/esd_max
† _refine_ls_shift/esd_mean
  _refine_ls_shift/su_max
  _refine_ls_shift/su_max_lt
  _refine_ls_shift/su_mean
  _refine_ls_shift/su_mean_lt
  _refine_ls_structure_factor_coef
  _refine_ls_weighting_details
```

```
  _refine_ls_weighting_scheme
  _refine_ls_wR_factor_all
  _refine_ls_wR_factor_gt
† _refine_ls_wR_factor_obs
  _refine_ls_wR_factor_ref
  _refine_special_details
```

(*b*) REFINE_LS_CLASS

```
• _refine_ls_class_code
        → _reflns_class_code
  _refine_ls_class_d_res_high
  _refine_ls_class_d_res_low
  _refine_ls_class_R_factor_all
  _refine_ls_class_R_factor_gt
  _refine_ls_class_R_Fsqd_factor
  _refine_ls_class_R_I_factor
  _refine_ls_class_wR_factor_all
```

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. The dagger (†) indicates a deprecated item, which should not be used in the creation of new CIFs.

Example 3.2.3.1 shows how the data names in the REFINE category are used. Most of the dictionary entries are detailed and fully explanatory, so only a few points that might require special care are mentioned here.

Two groups of older data names have been superseded by new names that are functionally equivalent, but represent a more correct terminology. One group is of names that include the component '**_obs**' used to indicate 'observed' reflections; this has been replaced by the component '**_gt**' indicating that the measured values are greater than a threshold recorded elsewhere (as the value of **_reflns_threshold_expression**). The other group replaces the component '**_esd**' (for estimated standard deviation) with '**_su**' (for standard uncertainty).

A number of data names describe the extinction coefficient and the method used to determine it. Note that a default value (Zachariasen) is given in the dictionary for the method (**_refine_ls_extinction_method**); this *only* makes sense if this data item is missing from the data block but a value of **_refine_ls_extinction_coef** is present. This can complicate the design of software to read CIFs, which might assign to any missing data name a default value given by the dictionary.

Care is also needed with **_refine_ls_hydrogen_treatment**, which describes the treatment of hydrogen atoms in the refinement. Clearly, the data item only has meaning if there were hydrogen atoms in the model (although, since in this case the default value is undef for 'undefined', it could be argued that the default is appropriate even when hydrogen atoms were not included in the model).

The weighting scheme used in the refinement is described by the two data names **_refine_ls_weighting_scheme** and **_refine_ls_weighting_details**. The first of the two can take only one of the three values sigma (weights assigned based on measured standard uncertainties), unit (unit or no weights applied) or calc (calculated weights applied). The actual mathematical expression used in the weighting scheme should be stated in **_refine_ls_weighting_details**.

A wide variety of 'residual structure-factor difference measures', referred to as *R* factors, are used in crystallography as indicators of refinement quality. The core CIF dictionary contains definitions for the three most commonly used *R* factors. The 'conventional *R* factor' is defined as

$$R = \frac{\sum |F(\text{meas.}) - F(\text{calc.})|}{\sum |F(\text{meas.})|},$$

Example 3.2.3.1. *Summary of refinement results.*

```
_refine_special_details
                sfls:_F_calc_weight_full_matrix

_refine_ls_structure_factor_coef  F
_refine_ls_matrix_type            full
_refine_ls_weighting_scheme       calc
_refine_ls_weighting_details
                  'w=1/(u^2^(F)+0.0004F^2^)'
_refine_ls_hydrogen_treatment     refxyz
_refine_ls_extinction_method      Zachariasen
_refine_ls_extinction_coef        3514(42)
_refine_ls_extinction_expression
;Larson, A. C. (1970). "Crystallographic Computing",
 edited by F. R. Ahmed. Eq. (22) p. 292. Copenhagen:
 Munksgaard.
;
_refine_ls_abs_structure_details
;The absolute configuration was assigned to agree
 with that of its precursor l-leucine at the
 chiral centre C3.
;
_refine_ls_number_reflns          1408
_refine_ls_number_parameters      272
_refine_ls_number_restraints      0
_refine_ls_number_constraints     0
_refine_ls_R_factor_all           .038
_refine_ls_R_factor_gt            .034
_refine_ls_wR_factor_all          .044
_refine_ls_wR_factor_gt           .042
_refine_ls_goodness_of_fit_all   1.462
_refine_ls_goodness_of_fit_gt    1.515
_refine_ls_shift/su_max           .535
_refine_ls_shift/su_mean          .044
_refine_diff_density_min         -.108
_refine_diff_density_max          .131
```

where $F(\text{meas.})$ and $F(\text{calc.})$ are the measured and calculated structure factors, respectively. In the data item **_refine_ls_R_factor_all**, the sum used in the calculation is taken over all the reflections collected, whereas in the data item **_refine_ls_R_factor_gt**, the sum is taken over reflections with a value greater than the limit specified by **_refine_threshold_expression**. In both cases, the reflections included in the calculation may be limited to those between specified resolution limits.

This *R* factor is calculated from the F values, regardless of whether the structure-factor coefficient $|F|$, $|F|^2$ or I was actually used in the refinement, and is often taken as a convenient indicator of the relative quality of a structure determination. As most structure refinements used to be performed on $|F|$, it allows a structure determined today to be compared with an older study.

Many refinements are now carried out on $|F|^2$, although some may still use the absolute value of the structure factor $|F|$ or the net intensity I. The weighted residual factor *wR* and goodness of fit *S* for a refinement should be reported according to the coefficients actually used in the refinement. For example, the weighted residual over all reflections, **_refine_ls_wR_factor_all**, is defined as

$$wR = \left(\frac{\sum w[Y(\text{meas.}) - Y(\text{calc.})]^2}{\sum wY(\text{meas.})^2} \right)^{1/2},$$

where w represents the weights and Y represents the structure-factor coefficient, either $|F|$, $|F|^2$ or I as specified by **_refine_ls_structure_factor_coef**.

This distinction between the conventional *R* factor, which is invariably calculated using F values, and the *wR* and *S* factors also holds for similar expressions defined on subsets of the reflections, *e.g.* **_reflns_class_wR_factor_all**.

Note that data names are also provided for reporting *unweighted* residuals on $|F|^2$ or I, but these are rarely used in practice, with

Example 3.2.3.2. *Structure-factor listing.*

```
loop_
     _refln_index_h
     _refln_index_k
     _refln_index_l
     _refln_F_squared_calc
     _refln_F_squared_meas
     _refln_F_squared_sigma
     _refln_include_status
    2   0   0       85.57       58.90      1.45 o
    3   0   0    15718.18    15631.06     30.40 o
    4   0   0    55613.11    49840.09     61.86 o
    5   0   0      246.85      241.86     10.02 o
    6   0   0       82.16       69.97      1.93 o
    7   0   0     1133.62      947.79     11.78 o
    8   0   0     2558.04     2453.33     20.44 o
    9   0   0      283.88      393.66      7.79 o
    10  0   0      283.70      171.98      4.26 o
```

the exception of $R(I)$ in Rietveld refinements against powder data, where it is generally called the Bragg R factor, $R_{\rm Bragg}$ or R_B.

The data items in the REFINE_LS_CLASS category are similar to several in the general REFINE category, but correspond to values for separate reflection classes as described in the REFLNS_CLASS category. The data name **_refine_ls_class_code** identifies the individual classes through a direct match with a corresponding value of **_reflns_class_code**.

3.2.3.2. Reflection measurements

The categories describing the reflections used in the refinement are as follows:

REFLN group

Individual reflections (§3.2.3.2.1)

REFLN

Groups of reflections (§3.2.3.2.2)

REFLNS

REFLNS_CLASS

REFLNS_SCALE

REFLNS_SHELL

The main category in this group is REFLN, which stores the list of reflections used in the structure refinement process, their associated structure factors and information about how each reflection was handled. The distinction between the REFLN (singular) category and the REFLNS (plural) category parallels the distinction between the categories DIFFRN_REFLN and DIFFRN_REFLNS: data items in the REFLN category store information about individual reflections, while data items in the REFLNS category store information about the complete set of reflections, or about subsets of reflections selected by shells of resolution, scaling factors or other criteria.

3.2.3.2.1. *Individual reflections*

The data items in this category are as follows:

REFLN

- **_refln_index_h**
- **_refln_index_k**
- **_refln_index_l**

_refln_A_calc
_refln_A_meas
_refln_B_calc
_refln_B_meas
_refln_class_code
→ **_reflns_class_code**
_refln_crystal_id
→ **_exptl_crystal_id**
_refln_d_spacing
_refln_F_calc
_refln_F_meas
_refln_F_sigma
_refln_F_squared_calc
_refln_F_squared_meas
_refln_F_squared_sigma
_refln_include_status
_refln_intensity_calc
_refln_intensity_meas
_refln_intensity_sigma
_refln_mean_path_length_tbar
† **_refln_observed_status**
_refln_phase_calc
_refln_phase_meas
_refln_refinement_status
_refln_scale_group_code
→ **_reflns_scale_group_code**
_refln_sint/lambda
_refln_symmetry_epsilon
_refln_symmetry_multiplicity
_refln_wavelength
_refln_wavelength_id
→ **_diffrn_radiation_wavelength_id**

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. The dagger (†) indicates a deprecated item, which should not be used in the creation of new CIFs.

Example 3.2.3.2 shows a typical structure-factor listing produced by a refinement program. This kind of structure-factor listing is suitable for deposition with a journal or a database. The Miller indices for each reflection are accompanied by the calculated and measured values of the quantity used in the refinement, and the standard uncertainty derived from the measurement. There is also an indication of whether each reflection was included in the refinement and in the calculation of R factors.

In this example, the squared structure factors $|F|^2$ are listed. When refinement is performed against the structure factors F or the intensities I, the data items **_refln_F_calc** or **_refln_intensity_calc** and the corresponding data names for the measured values and uncertainties should be used.

Individual calculated structure-factor components $A = |F|\cos\varphi$ and $B = |F|\sin\varphi$ may also be listed, along with the phase φ, using the data names **_refln_A_calc**, **_refln_B_calc** and **_refln_phase_calc**. Corresponding measured values have equivalent ***_meas** names.

The **_refln_include_status** flag is used to indicate whether reflections were used in the refinement and in the calculation of R factors, and if they were not used, to give the reason for exclusion of the reflection from the refinement. The flag o, which indicates that a reflection was used in the refinement, was originally chosen to indicate that the value of the reflection was higher than the limit specified by **_reflns_observed_criterion** and that the reflection was thus 'observed'. The data item **_reflns_observed_criterion** is now deprecated in favour of **_reflns_threshold_status**, and the value o is now taken to indicate not only that the reflection has an intensity suitable for inclusion in the refinement, but also that the reflection satisfies all other criteria used to select reflections for inclusion in the refinement.

Various other flags indicate reflections that were not included in the refinement. Reflections outside the range of d spacings bounded by the values **_refine_ls_d_res_high** and **_refine_ls_d_res_low** are flagged with h or l, respectively. Reflections within the resolution limits but below the intensity threshold are flagged with <. Systematically absent reflections are flagged with -. Sometimes a value can be identified as having a systematic error; these reflections can be flagged with x. However, great care must be taken in excluding reflections with apparently 'anomalous' structure factors (*i.e.* where the measured values are substantially different from the calculated ones), so as not to introduce bias into the refinement.

The flag **_refln_refinement_status** is used specifically to indicate whether a reflection was included in or excluded from

the refinement. Use of **_refln_include_status** to provide more information about each reflection is greatly preferred.

Other data names in this category allow the recording of specific information about each reflection, such as the symmetry reinforcement factor ε, the number of reflections symmetry-equivalent under the Laue symmetry, the d spacing, the mean path length through the crystal $\bar{t}$, the $(\sin\theta)/\lambda$ value and, in the case of Laue experiments, the mean wavelength of the radiation. (For polychromatic radiation, the wavelength information might instead be given by **_refln_wavelength_id**, which is a code identifying a matching entry in the DIFFRN_RADIATION category.)

Other codes provide links to identifiers in other categories. The **_refln_class_code** identifies a set of reflections binned as described by entries in the REFLNS_CLASS category. **_refln_scale_group_code** identifies groups of reflections to which the same structure-factor scaling has been applied.

Note that the values of the Miller indices in this list must correspond to the cell defined by the lengths and angles recorded in the CELL category; they may, however, be different from the Miller indices in the DIFFRN_REFLN list if a transformation of the original cell has taken place. In this case, the transformation matrix is given using the **_diffrn_reflns_transf_matrix_*** items.

The usual use of a CIF as an archive of a completed structure determination implies that the values given in the REFLN list are derived from the final cycle of refinement, but this is not a formal requirement. Care should be taken when preparing a CIF for archiving that the structural model corresponds to the refinement cycle summarized in the accompanying REFLN table, especially if the file is constructed from fragments output from different programs.

3.2.3.2.2. *Groups of reflections*

The data items in these categories are as follows:

(*a*) REFLNS

```
  _reflns_d_resolution_high
  _reflns_d_resolution_low
  _reflns_Friedel_coverage
  _reflns_limit_h_max
  _reflns_limit_h_min
  _reflns_limit_k_max
  _reflns_limit_k_min
  _reflns_limit_l_max
  _reflns_limit_l_min
  _reflns_number_gt
† _reflns_number_observed
  _reflns_number_total
† _reflns_observed_criterion
  _reflns_special_details
  _reflns_threshold_expression
```

(*b*) REFLNS_CLASS

```
• _reflns_class_code
  _reflns_class_d_res_high
  _reflns_class_d_res_low
  _reflns_class_description
  _reflns_class_number_gt
  _reflns_class_number_total
  _reflns_class_R_factor_all
  _reflns_class_R_factor_gt
  _reflns_class_R_Fsqd_factor
  _reflns_class_R_I_factor
  _reflns_class_wR_factor_all
```

(*c*) REFLNS_SCALE

```
• _reflns_scale_group_code
  _reflns_scale_meas_F
  _reflns_scale_meas_F_squared
  _reflns_scale_meas_intensity
```

(*d*) REFLNS_SHELL

```
• _reflns_shell_d_res_high
• _reflns_shell_d_res_low
† _reflns_shell_meanI_over_sigI_all
† _reflns_shell_meanI_over_sigI_gt
† _reflns_shell_meanI_over_sigI_obs
  _reflns_shell_meanI_over_uI_all
  _reflns_shell_meanI_over_uI_gt
  _reflns_shell_number_measured_all
  _reflns_shell_number_measured_gt
† _reflns_shell_number_measured_obs
  _reflns_shell_number_possible
  _reflns_shell_number_unique_all
  _reflns_shell_number_unique_gt
† _reflns_shell_number_unique_obs
  _reflns_shell_percent_possible_all
  _reflns_shell_percent_possible_gt
† _reflns_shell_percent_possible_obs
  _reflns_shell_Rmerge_F_all
  _reflns_shell_Rmerge_F_gt
† _reflns_shell_Rmerge_F_obs
  _reflns_shell_Rmerge_I_all
  _reflns_shell_Rmerge_I_gt
† _reflns_shell_Rmerge_I_obs
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The dagger (†) indicates a deprecated item, which should not be used in the creation of new CIFs.

The data items in the REFLNS category describe properties or attributes of the complete set of reflections used in the structure refinement. Several are derivative and may be obtained from the information in the reflections list, but it is convenient to present them separately so that they do not need to be calculated again. They can also be used to check the consistency of the reflections list.

The **_reflns_limit_*** data items define the upper and lower bounds on the Miller indices and on the interplanar d spacings.

The **_reflns_threshold_expression** is a text field describing the criterion applied to mark individual reflections as 'significantly intense' (*i.e.* distinct from the background level). This is typically expressed as a multiple of the standard uncertainty on the quantity used in refinement, *e.g.* `I>2u(I)`.

The number of reflections with values higher than the threshold is reported in **_reflns_number_gt**. The total number of reflections measured is given by **_reflns_number_total**. Although the use of these data names appears to be obvious, different practices have been used in the past to report total numbers (*e.g.* by neglecting symmetry-equivalent reflections) and the definitions in the dictionary should be consulted. Both numbers may contain Friedel-equivalent reflections (those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class).

The proportion of Friedel-related reflections present is reported separately by **_reflns_Friedel_coverage**, defined as $(N_C - N_L)/N_L$, where N_C is the number of reflections obtained on averaging under the symmetry of the crystal class and N_L is the number obtained on averaging under the Laue class. The definition in the dictionary provides examples of how the value of this data name may be used as an indicator of the fraction of the available reciprocal space sampled in the diffraction experiment.

The deprecated data names **_reflns_observed_criterion** and **_reflns_number_observed** reflect the old use of 'observed' as a term describing significantly intense reflections. They should not be used in the creation of new CIFs, but are retained to ensure that the information can be extracted from old CIFs.

The free-text field **_reflns_special_details** can be used to discuss any aspects of the reflections list not covered by other data names. It is recommended that information about the averaging of symmetry-equivalent reflections (including Friedel pairs) should be given here.

Example 3.2.3.3. *Description of subsets of the reflection list.*

```
loop_
  _reflns_class_number_gt
  _reflns_class_code
  _reflns_class_description
      584    'Main'   'm=0; main reflections'
      226    'Sat1'   'm=1; first-order satellites'
      50     'Sat2'   'm=2; second-order satellites'
```

The REFLNS_CLASS category is used to summarize the properties of subsets of the reflection list. The data names are analogous to several in the REFLNS and REFINE categories, but are applied to individual classes of reflections labelled by **_reflns_class_code** and described by **_reflns_class_description** (see Example 3.2.3.3).

Individual reflections in the structure-factor listing can be recognized through the matching value of **_refln_class_code** as belonging to a particular class labelled by **_reflns_class_code**.

Although classes can be assigned according to arbitrary criteria, the specific case for which the REFLNS_CLASS category was designed was the partitioning of the reflection list into contributions from different components in incommensurately modulated structures. However, the formalism is general and other binning strategies can be described. Note, however, that the specific case of processing of reflections by shells of resolution (in macromolecular crystallography, for example) is handled explicitly by the REFLNS_SHELL category.

The category REFLNS_SCALE provides a listing of the scale factors applied to individual reflections sharing a common value of **_refln_scale_group_code**. Each value is indexed by the matching identifier **_reflns_scale_group_code** of this category.

The REFLNS_SHELL category describes the properties of separate resolution shells of reflections and is a special case of the binning of reflections into classes (compare REFLNS_CLASS above).

Each shell is defined by an upper and lower resolution limit (**_reflns_shell_d_res_high** and ***_low**), and for each shell there are data names for the number of reflections measured and exceeding a threshold of significance, for the percentage of geometrically possible reflections collected, and for the ratios of the mean intensities to their standard uncertainties.

$R_{\rm merge}$ values are also defined for each shell of resolution (both for all measured reflections and for significantly intense ones).

This category also contains a number of deprecated data names reflecting older terminology and notation. Such data names should not be used in creating new CIFs, but will need to be recognized by CIF-reading software in order to process old CIFs.

3.2.4. Atomicity, chemistry and structure

The core CIF dictionary provides many data names for describing the structural model.

The categories describing the atom sites handle these in a general way as sites of significant electron density which might be contributed to by more than one element species. The chemical identification of the compound under study, and where appropriate a model of the molecular connectivity and bonding, are handled separately by the chemistry-related categories. The geometry-related categories are purely derivative, given knowledge of the positions of the atom sites and the crystallographic symmetry; but as with other examples of derived data, they are given their own data names to provide convenient listings and to check the consistency of information provided by other categories. The symmetry-related data names in the core dictionary are restricted to those essential for the construction of a geometric model; Chapter 3.8 describes a symmetry extension dictionary suitable for a more complete description of crystal symmetry.

3.2.4.1. Atom sites

The categories describing atom sites are as follows:

ATOM group

Individual atom sites (§3.2.4.1.1)

ATOM_SITE

Collections of atom sites (§3.2.4.1.2)

ATOM_SITES

Atom types (§3.2.4.1.3)

ATOM_TYPE

These categories permit the traditional interpretation of regular concentrations of electron density in a crystalline lattice as atom sites containing one or more chemical elements, with complete or partial occupancy, and with a spatial distribution affected by thermal displacement or disorder.

Lists of atom-site coordinates and anisotropic displacement factors are covered by data items in the ATOM_SITE category. Identification of the chemical species occupying each site is handled by data items in the ATOM_TYPE category and data items in the ATOM_SITES category record collective information common to all sites.

While the ATOM_SITE category formally contains the data items describing both positions and atomic displacements, the anisotropic displacement parameters are often given in a separate looped list. In the version of the core dictionary embedded in the macromolecular CIF dictionary, which uses the DDL2 formalism, this is recognized by the creation of a separate, but overlapping, ATOM_SITE_ANISOTROP category.

3.2.4.1.1. *Individual atom sites*

The data items in this category are as follows:

ATOM_SITE

```
• _atom_site_label
  _atom_site_adp_type
  _atom_site_aniso_B_11
  _atom_site_aniso_B_12
  _atom_site_aniso_B_13
  _atom_site_aniso_B_22
  _atom_site_aniso_B_23
  _atom_site_aniso_B_33
  _atom_site_aniso_label
        → _atom_site_label
  _atom_site_aniso_ratio
  _atom_site_aniso_type_symbol
        → _atom_site_type_symbol
  _atom_site_aniso_U_11
  _atom_site_aniso_U_12
  _atom_site_aniso_U_13
  _atom_site_aniso_U_22
  _atom_site_aniso_U_23
  _atom_site_aniso_U_33
  _atom_site_attached_hydrogens
  _atom_site_B_equiv_geom_mean
  _atom_site_B_iso_or_equiv
  _atom_site_calc_attached_atom
  _atom_site_calc_flag
  _atom_site_Cartn_x
  _atom_site_Cartn_y
  _atom_site_Cartn_z
  _atom_site_chemical_conn_number
        → _chemical_conn_atom_number
  _atom_site_constraints
  _atom_site_description
  _atom_site_disorder_assembly
  _atom_site_disorder_group
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_label_component_0
  _atom_site_label_component_1
```

```
  _atom_site_label_component_2
  _atom_site_label_component_3
  _atom_site_label_component_4
  _atom_site_label_component_5
  _atom_site_label_component_6
  _atom_site_occupancy
† _atom_site_refinement_flags
  _atom_site_refinement_flags_adp
  _atom_site_refinement_flags_occupancy
  _atom_site_refinement_flags_posn
  _atom_site_restraints
  _atom_site_symmetry_multiplicity
† _atom_site_thermal_displace_type
  _atom_site_type_symbol
        → _atom_type_symbol
  _atom_site_U_equiv_geom_mean
  _atom_site_U_iso_or_equiv
  _atom_site_Wyckoff_symbol
```

The bullet (•) indicates a category key. For this category an alternative category key can be formed by taking all the **_atom_site_label_component_*** *items together. Anisotropic displacement parameters may also be listed in a separate loop, for which* **_atom_site_aniso_label** *forms the key. The arrow (→) is a reference to a parent data item. The dagger (†) indicates a deprecated item, which should not be used in the creation of new CIFs.*

Data items in the ATOM_SITE category represent the positions of atom sites identified in the structural model, their spatial distribution defined by isotropic or anisotropic displacement parameters, details of restraints or constraints applied during the refinement, and the interpretation of their occupancy due to structural or compositional disorder.

Example 3.2.4.1 is a typical extract from a list of atom-site coordinates, with equivalent isotropic displacement values and refinement conditions. Each site is identified by **_atom_site_label**.

The coordinates are specified as fractional x, y, z values along the unit-cell axes. Coordinates may also be specified in ångström units along orthogonal Cartesian axes using the data names **_atom_site_Cartn_x**, **_atom_site_Cartn_y** and **_atom_site_Cartn_z**. The transformation matrix between Cartesian and fractional coordinates can be given in the ATOM_SITES category.

(Note that occupancy values are unaffected by symmetry. This is discussed later in connection with site multiplicity.)

_atom_site_U_iso_or_equiv records the isotropic atomic displacement value $U_{\rm iso}$ in the case of isotropic refinement. In the case of anisotropic refinement, **_atom_site_U_iso_or_equiv** records the equivalent isotropic value $U_{\rm eq}$, defined as

$$U_{\rm eq} = (1/3)\sum_i \left[\sum_j (U^{ij} a_i^* a_j^* a_i a_j)\right],$$

where a_i are the real-space cell lengths, a_j^* are the reciprocal-space cell lengths and U^{ij} are the anisotropic displacement parameters.

The data item **_atom_site_adp_type** identifies which value is given. An alternative equivalent isotropic displacement parameter **_atom_site_U_equiv_geom_mean** may be calculated as the geometric mean of the anisotropic parameters,

$$U_{\rm eq} = (U_i U_j U_k)^{1/3},$$

where the U_i are the principal components of the orthogonalized U^{ij}.

Data names also exist for the corresponding quantities calculated from B values, although the use of B values is discouraged by the IUCr Commission on Crystallographic Nomenclature.

For each site, **_atom_site_calc_flag** takes one of the following values: `d`, to indicate that the atom-site coordinates were determined from the diffraction intensities; `c` or `calc` to indicate that they were calculated from molecular geometry considerations; or `dum`, for a dummy site.

Example 3.2.4.1. *List of atom-site coordinates, equivalent isotropic U values and refinement conditions.*

```
loop_
     _atom_site_label
     _atom_site_fract_x
     _atom_site_fract_y
     _atom_site_fract_z
     _atom_site_U_iso_or_equiv
     _atom_site_adp_type
     _atom_site_calc_flag
     _atom_site_refinement_flags_posn
     _atom_site_occupancy
     _atom_site_disorder_assembly
     _atom_site_disorder_group
     _atom_site_type_symbol
   O1 0.5000 1.0000 0.8011(2) 0.0259(6)
      Uani d    S 1 . . O
   O2 0.33569(10) 0.98239(10) 0.88892(17) 0.0321(5)
      Uani d    . 1 . . O
   O3 0.20150(13) 0.92560(11) 0.8817(2) 0.0458(5)
      Uani d    . 1 . . O
   O4 0.35539(11) 0.81530(10) 0.96958(17) 0.0333(5)
      Uani d    . 1 . . O
   C1 0.43883(16) 0.95672(14) 0.7293(3) 0.0275(6)
      Uani d    . 1 . . C
   H1 0.4064 0.9930 0.6730 0.0493(19)
      Uiso calc R 1 . . H
   C2 0.37292(16) 0.92010(14) 0.8175(3) 0.0266(6)
      Uani d    . 1 . . C
   H2 0.3246 0.8945 0.7691 0.0493(19)
      Uiso calc R 1 . . H
   C3 0.41827(16) 0.85968(14) 0.8983(3) 0.0280(6)
      Uani d    . 1 . . C
   H3 0.4629 0.8850 0.9536 0.0493(19)
      Uiso calc R 1 . . H
```

Specific restraints or constraints applied to a site may be indicated by one or more of the **_atom_site_refinement_flags_*** items.

The data item **_atom_site_occupancy** defines the fraction of the atom type present at the site. Note that the same site may occur more than once in the list, identified by separate values of **_atom_site_label**. Such an arrangement would represent contributions from separate atom types (perhaps in modelling compositional disorder). The sum of occupancies of all atom types present at a single site may not significantly exceed 1.0 (unless it is a dummy site with no physical significance). Note that an atom of a given chemical species positioned on a special position (*e.g.* on a twofold axis) will in general be assigned a full occupancy value of 1.0. However, it will occur less often in the unit cell than an atom on a general position (in this example by a factor of 2). To account for this in structure-factor calculations it may be given a population value of 0.5 within the refinement program. A population adjustment of this kind is *not* implied in the assignment of a value to **_atom_site_occupancy**. The multiplicity of the site owing to the space-group symmetry can be recorded in **_atom_site_symmetry_multiplicity**.

The disorder-related data names in this example will be discussed below.

_atom_site_type_symbol is a code which must match an entry in the ATOM_TYPE category that supplies information about the elemental composition and scattering factors of the atom or atoms occupying the site. Note that it is quite legitimate to have an atom-type symbol such as 'Fe3+Ni2+', referring to a mixed-composition atom site. The effective physical properties of such a pseudo-atom should be given in full in the ATOM_TYPE category.

Example 3.2.4.2 demonstrates how the anisotropic displacement parameters are conventionally broken out into a separate list. When this is done, each atom site is identified by **_atom_site_aniso_label**, and this must of course match the value of **_atom_site_label** specifying the position of the site.

Example 3.2.4.2. *Separate list of anisotropic U values with* **_atom_site_aniso_label** *acting as the key that uniquely identifies table rows in this listing.*

```
loop_
     _atom_site_aniso_label
     _atom_site_aniso_U_11
     _atom_site_aniso_U_22
     _atom_site_aniso_U_33
     _atom_site_aniso_U_12
     _atom_site_aniso_U_13
     _atom_site_aniso_U_23
    O1 0.0256(9) 0.0275(9) 0.0433(12) 0.0020(8)
       0.0038(8) -0.0062(9)
    O2 0.0306(10) 0.0418(11) 0.0651(15) -0.0037(10)
       0.0079(11) -0.0049(11)
    O3 0.0374(10) 0.0309(10) 0.0315(11) -0.0030(9)
       0.0051(9) 0.0005(9)
    C1 0.0279(13) 0.0245(13) 0.0300(15) 0.0004(11)
       -0.0020(12) -0.0034(12)
    C2 0.0258(13) 0.0226(13) 0.0314(15) 0.0024(11)
       -0.0024(12) -0.0033(12)
    C3 0.0292(13) 0.0246(13) 0.0302(16) -0.0028(11)
       0.0031(12) 0.0005(12)
```

The data item **_atom_site_label** is normally used as the identifier of each individual atom site in a list of coordinates and atomic displacement factors. Historically, the labels given to atom sites have been chosen to summarize useful information about the atom located at the site. Almost invariably the label contains the symbol of the chemical element or elements occupying the site; there may also be indicators of charge, valence, chemical connectivity, disorder, occupation of a site of crystallographic symmetry or grouping within a component of secondary structure within large molecules. In a CIF, it is formally sufficient that atom-site labels are unique, as all the information about composition, valence, connectivity and so on can be extracted from the data items designed specifically to record this information. However, it is preferable that an atom-site label should summarize the relevant features of the site. Many styles and conventions for labelling atoms are in use in crystallography, so to enable interchange with other crystallographic data file formats, the core dictionary contains a detailed but highly flexible set of rules for constructing and parsing atom-site labels.

Labelling atom sites in crystallography usually serves two distinct purposes: (*a*) to identify the site in the molecule and crystal, and (*b*) to identify the chemical element that occupies that site. The core dictionary makes this distinction clear by defining ATOM_SITE and ATOM_TYPE as separate data categories. The connection between the two categories is made through the equivalence of the data items **_atom_site_type_symbol** (in the ATOM_SITE list) and **_atom_type_symbol** (in the ATOM_TYPE list). Often, however, crystallographers use a single label **_atom_site_label** to define both the site and the chemical species occupying it.

The **_atom_site_label** may be composed of as many as eight separate components; the recommended convention for construction of the string is as follows.

Component 0 [optionally identical to a value of **_atom_type_symbol**] (*mandatory*): A character string containing any character except a blank or an underline, with the proviso that each digit '0'–'9' is used only to designate an oxidation state and, as such, must be followed by a plus '+' or a minus '-' character. It is recommended that the element symbols be used when applicable. Examples of permissible codes are: `Cu`, `Cu2+`, `dummy`, `Fe3+Ni2+`, `S-`, `H*`, `H(SDS)`.

Component 1 [atom number code] (*optional*): This string may contain any alphanumeric character except a blank or an underline, but the first character *must* be a digit '0'–'9' and the second character may not be a plus '+' or a minus '-'. Component 1 is intended primarily to differentiate sites containing the same atom type, but it can be used for any purpose. Examples of combined component 0 and 1 codes are: **C**`1`, **C**`103g28`, **Fe3+**`17b`, **H***`251`, **boron**`2a`, **Ni2+**`2`, **Fe2+Ni2+**`2`, where component 0 is in bold to indicate how these labels are parsed.

Component 2 [residue code] (*optional*): This string may contain any character except a blank or underline. It is intended primarily to give specific structural information such as the molecular fragment or amino-acid type, *e.g.* `C1_gly`, `O1_SO4`. If component 2 is present, it is separated from the concatenated components 0 and 1 with an underline character.

Components 3–7 [sequence, remoteness, chain order, alternate, footnote codes] (*optional*): These strings may contain any character except a blank or an underline. The underline character is used to separate the individual components. The names associated with the separate components suggest their roles in constructing composite labels that match the conventions of site labelling in the PDB format for macromolecular structure files. However, they are not restricted to these functions and may be used in other ways.

Component 0 is normally identical to an **_atom_type_symbol** code in the ATOM_TYPE list. However, if it is not, an **_atom_site_type_symbol** code must appear in the ATOM_TYPE list in order to identify the atom type. In these cases, component 0 may contain any code consistent with the rules given in the dictionary. Thus, component 0 could be `Ca` to identify an alpha carbon, provided that the **_atom_site_type_symbol** is encoded as `C` to indicate that the atom type is carbon.

Multiple occupation of a single atom site by different atom species (compositional disorder) may be handled simply by having multiple values of **_atom_site_label** referring to the same site in the crystal structure. Alternatively, multiple occupancy of an atom site may be denoted by a unique character or characters in component 0 of the atom label, with the ATOM_TYPE list containing the equivalent pseudo element label entry with values that are weighted averages of those for the constituent elements. The proportions of the atom types should then be defined using **_atom_type_description**.

This **_atom_site_label** construction is flexible, visually decipherable and well suited to computer applications. The components can be easily identified and stripped with a single pass, from left to right, along the label string. Note that the underline separators are only used if higher-order components exist. If intermediate components are not used they may be omitted provided the underline separators are retained. For example, the label `C233__ggg` is acceptable and contains the components 0: `C`, 1: `233`, 2: *null* and 3: `ggg`. There is no requirement that the same number of components should be used in each label.

The **_atom_site_label** may be replaced by separate data items specifying the individual components of an atom label; this may be useful for large lists of site coordinates, for example in a macromolecular structure, where site-labelling components follow a systematic convention and where subsets of the atom sites need to be searched for or extracted using individual label components. Such uses are not common in files built with core CIF data names; the mmCIF dictionary identifies substructural components in biological macromolecules by alternative techniques (Section 3.6.7).

There is no comparable fragmentation of the components of **_atom_site_aniso_label**. Where separate lists of anisotropic displacement parameters use complex atom-site labels, either the coordinate list should use **_atom_site_label** alone or the processing software needs to be able to construct a value for **_atom_site_label** from the separate components **_atom_site_label_component_*** in order to test the equivalence between the

Example 3.2.4.3. *Chemical connectivity table; atoms are linked back to atom-site positions through matching values of* **_atom_site_chemical_conn_number** *and* **_chemical_conn_atom_number**.

```
loop_
  _atom_site_label
  _atom_site_chemical_conn_number
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
  S1 1 0.74799(9)   -0.12482(11) 0.27574(9) 0.0742(3)
  S2 2 1.08535(10)   0.16131(9)  0.34061(9) 0.0741(3)
  N1 3 1.0650(2)    -0.1390(2)   0.2918(2)  0.0500(5)
  C1 4 0.9619(3)    -0.0522(3)   0.3009(2)  0.0509(6)
     # - - - - data truncated for brevity - - - -

loop_
  _chemical_conn_atom_number
  _chemical_conn_atom_type_symbol
  _chemical_conn_atom_display_x
  _chemical_conn_atom_display_y
  _chemical_conn_atom_NCA
  _chemical_conn_atom_NH
1   S    .39  .81   1   0   2   S    .39  .96   2   0
3   N    .14  .88   3   0   4   C    .33  .88   3   0
5   C    .11  .96   2   2   6   C    .03  .96   2   2
     # - - - - data truncated for brevity - - - -
```

Example 3.2.4.4. *Handling of occupational disorder of atom sites.*

```
loop_
  _atom_site_label
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_occupancy
  _atom_site_disorder_assembly
  _atom_site_disorder_group
    B2    0.9639(7)  0.6536(5)  0.4464(5)  1     .  .
    F21   0.9411(18) 0.7083(11) 0.5388(9)  0.50  A  1
    F22   1.008(2)   0.5331(6)  0.4747(10) 0.50  A  1
    F23   0.8364(14) 0.6845(17) 0.3951(15) 0.50  A  1
    F24   1.0718(17) 0.6896(14) 0.3764(12) 0.50  A  1
    F21A  0.9727(18) 0.7141(15) 0.5293(13) 0.50  A  2
    F22A  1.0540(15) 0.5401(8)  0.4635(15) 0.50  A  2
    F23A  0.8216(9)  0.6479(15) 0.4461(14) 0.50  A  2
    F24A  1.007(2)   0.7096(17) 0.3476(11) 0.50  A  2
```

labels in the coordinates and anisotropic displacement parameters lists.

While either atom-labelling technique is permitted, it is recommended that the individual label components are *not* used unless there is an overwhelming argument to do so.

Information about the molecular model is sometimes embedded in a labelling convention. In CIF, this information is usually expressed through other data items.

The connectivity of a molecule is described by the CHEMICAL group of categories, and more specifically through the CHEMICAL_CONN_ATOM and CHEMICAL_CONN_BOND categories.

The link between atom sites in the coordinate list and the corresponding atoms in the molecular model is established using the data item **_chemical_conn_atom_number** in the CHEM_CONN_ATOM category, and the data items **_chemical_conn_bond_atom_1** and **_chemical_conn_bond_atom_2** in the CHEMICAL_CONN_BOND category. The values of these data items must match values for the data item **_atom_site_chemical_conn_number** in the ATOM_SITE list. Example 3.2.4.3 shows an extract from a connectivity table; a more complete version of this table is given in the relevant category descriptions in the dictionary.

Note that there is no guarantee that the refined atom-site coordinates that characterize the asymmetric unit will correspond to locations within a single connected molecular species. Crystal symmetry transformations may need to be applied to individual sites in order to map the contents of a connected molecular residue to real space in the unit cell. There is no provision in the CHEMICAL_CONN categories for the specification of these symmetry transformations; thus these higher-order molecular geometries are best described using data items in the GEOM categories, which do allow for the specification of symmetry transformations.

It may also be the case that not all atom positions have been located; this is particularly true for hydrogen atoms, and the data item **_atom_site_attached_hydrogens** is provided for bookkeeping purposes to indicate hydrogen atoms known to be bonded to an atom but whose positions have not been refined (or calculated).

Example 3.2.4.4 shows how the disorder of a group of bonded atoms over a set of atom sites (occupational disorder) is described. In this example of a disordered tetrafluoroborate anion, the data item **_atom_site_disorder_assembly** takes the value A, and the data item **_atom_site_disorder_group** takes the values 1 and 2, indicating the two alternative positions of the disordered group.

The remaining items in this category are clearly described in their individual dictionary entries. However, the now-deprecated data item **_atom_site_refinement_flags** should be mentioned. This was allowed to take values obtained by concatenating one or more of the single-letter flags:

- . no refinement constraints;
- S special-position constraint on site;
- G rigid-group refinement of site;
- R riding-atom site attached to non-riding atom;
- D distance or angle restraint on site;
- T thermal displacement constraints;
- U $U_{\rm iso}$ or U^{ij} restraint (rigid bond);
- P partial occupancy constraint.

These individual flags are listed in the dictionary using the DDL field **_enumeration**, which denotes a list of mutually exclusive permitted values. As concatenation of values is allowed here, dictionary-based software must be modified to handle this data item as a special case. To avoid the need for this in future, the data item was marked as deprecated from version 2.3 of the dictionary, and is replaced by the three separate items **_atom_site_refinement_flags_posn**, ***_adp** and ***_occupancy**. For each of these, the relevant combinations of refinement flags are fully enumerated (for example **_atom_site_refinement_flags_adp** may take any one of the values T, U or TU). This logically separates the different types of refinement constraints or restraints that an author might want to record and allows software to parse the data item.

3.2.4.1.2. *Collections of atom sites*

The data items in this category are as follows:

ATOM_SITES

```
_atom_sites_Cartn_tran_matrix_11
_atom_sites_Cartn_tran_matrix_12
_atom_sites_Cartn_tran_matrix_13
_atom_sites_Cartn_tran_matrix_21
_atom_sites_Cartn_tran_matrix_22
_atom_sites_Cartn_tran_matrix_23
_atom_sites_Cartn_tran_matrix_31
_atom_sites_Cartn_tran_matrix_32
_atom_sites_Cartn_tran_matrix_33
_atom_sites_Cartn_tran_vector_1
_atom_sites_Cartn_tran_vector_2
_atom_sites_Cartn_tran_vector_3
_atom_sites_Cartn_transform_axes
_atom_sites_fract_tran_matrix_11
_atom_sites_fract_tran_matrix_12
_atom_sites_fract_tran_matrix_13
```

```
_atom_sites_fract_tran_matrix_21
_atom_sites_fract_tran_matrix_22
_atom_sites_fract_tran_matrix_23
_atom_sites_fract_tran_matrix_31
_atom_sites_fract_tran_matrix_32
_atom_sites_fract_tran_matrix_33
_atom_sites_fract_tran_vector_1
_atom_sites_fract_tran_vector_2
_atom_sites_fract_tran_vector_3
_atom_sites_solution_hydrogens
_atom_sites_solution_primary
_atom_sites_solution_secondary
_atom_sites_special_details
```

This category records information that applies collectively to the atom sites of the structural model. At present, the topics covered are the transformation matrix between Cartesian and cell fractional coordinates, and the methods used to locate the initial atom sites. **_atom_sites_solution_primary** describes how the first atom sites were determined, **_atom_sites_solution_secondary** describes how the remaining non-hydrogen sites were located and **_atom_sites_solution_hydrogens** describes how hydrogen atoms were located. The codes that are allowed for each of these refer to distinct solution methods, and at present only the seven formal values listed below are provided (although other values might be added in the future):

`difmap`	difference-electron-density map;
`vecmap`	real-space vector search;
`heavy`	heavy-atom method;
`direct`	structure-invariant direct methods;
`geom`	inferred from neighbouring sites;
`disper`	anomalous-dispersion techniques;
`isomor`	isomorphous structure methods.

3.2.4.1.3. *Atom types*

The data items in this category are as follows:

ATOM_TYPE

- **_atom_type_symbol**

```
_atom_type_analytical_mass_%
_atom_type_description
_atom_type_number_in_cell
_atom_type_oxidation_number
_atom_type_radius_bond
_atom_type_radius_contact
_atom_type_scat_Cromer_Mann_a1
_atom_type_scat_Cromer_Mann_a2
_atom_type_scat_Cromer_Mann_a3
_atom_type_scat_Cromer_Mann_a4
_atom_type_scat_Cromer_Mann_b1
_atom_type_scat_Cromer_Mann_b2
_atom_type_scat_Cromer_Mann_b3
_atom_type_scat_Cromer_Mann_b4
_atom_type_scat_Cromer_Mann_c
_atom_type_scat_dispersion_imag
_atom_type_scat_dispersion_real
_atom_type_scat_dispersion_source
_atom_type_scat_length_neutron
_atom_type_scat_source
_atom_type_scat_versus_stol_list
```

The bullet (•) *indicates a category key.*

The data items in this category record details about the atomic species associated with each occupied atom site in the structural model. While these will typically be standard properties of the naturally occurring chemical elements, they may also be synthetic atom types, for example in cases where a single atom site may be occupied with partial occupancies by atoms of different elements.

As mentioned in Section 3.2.4.1.1, there are two ways of dealing with such a case: the same location in the coordinate list may be populated by multiple entries, each for an atom of a particular element with an associated occupancy fraction; or a single entry may be made for the synthetic atom type, the properties of which are described fully in the ATOM_TYPE list.

Example 3.2.4.5. *Reference to atomic scattering factors.*

```
loop_
  _atom_type_symbol
  _atom_type_oxidation_number
  _atom_type_number_in_cell
  _atom_type_scat_dispersion_real
  _atom_type_scat_dispersion_imag
  _atom_type_scat_source
   C  0  72  .017  .009
             International_Tables_Vol_IV_Table_2.2B
   H  0  100  0     0
             International_Tables_Vol_IV_Table_2.2B
   O  0  12  .047  .032
             International_Tables_Vol_IV_Table_2.2B
   N  0  4   .029  .018
             International_Tables_Vol_IV_Table_2.2B
```

Each different atom type has a unique **_atom_type_symbol** identifier. In principle, this could be any string of characters, but the dictionary recommends certain conventions to encourage compatibility with the atom-site labelling rules. It is recommended that the identifier be the normal chemical element symbol when the atom type is a pure element. If some other labelling is used, the identifier may be composed of any character except an underline, with the additional proviso that digits designate an oxidation state and must be followed by a '+' or '-' character.

The data item **_atom_type_scat_versus_stol_list** can be used to give a table of scattering factors as a function of $(\sin\theta)/\lambda$. This is a text field with no specified internal structure, except the suggestion that it is well commented and the lists should be regularly formatted. However, it is generally enough to list the atomic scattering factors of each element and to provide a reference to the source of the values, as in Example 3.2.4.5.

3.2.4.2. Chemical identification and connectivity information

The categories describing chemical identity and connectivity are as follows:

CHEMICAL group

Chemical identification (§3.2.4.2.1)

CHEMICAL

CHEMICAL_FORMULA

Chemical connectivity (§3.2.4.2.2)

CHEMICAL_CONN_ATOM

CHEMICAL_CONN_BOND

As indicated in Section 3.2.4.1.1, the chemical interpretation of the coordinate list of regions of significant electron density is not always easy. Occupational and compositional disorder, symmetry-equivalent locations, and unrefined atom sites all contribute to the difficulties, but it is usually possible in modern studies to construct a sensible chemical model. The CHEMICAL category group provides the data names needed to describe the chemical identity and properties of the material characterized in the structural study.

3.2.4.2.1. *Chemical identification*

The data items in these categories are as follows:

(*a*) CHEMICAL

```
_chemical_absolute_configuration
_chemical_compound_source
_chemical_melting_point
_chemical_melting_point_gt
_chemical_melting_point_lt
_chemical_name_common
_chemical_name_mineral
_chemical_name_structure_type
_chemical_name_systematic
```

```
_chemical_optical_rotation
_chemical_properties_biological
_chemical_properties_physical
_chemical_temperature_decomposition
_chemical_temperature_decomposition_gt
_chemical_temperature_decomposition_lt
_chemical_temperature_sublimation
_chemical_temperature_sublimation_gt
_chemical_temperature_sublimation_lt
```

(*b*) CHEMICAL_FORMULA

```
_chemical_formula_analytical
_chemical_formula_iupac
_chemical_formula_moiety
_chemical_formula_structural
_chemical_formula_sum
_chemical_formula_weight
_chemical_formula_weight_meas
```

The CHEMICAL category itself deals with the large-scale chemical properties of the compound from which the crystal under study was formed: its various formal and common names, its source, melting point, decomposition and sublimation temperatures (as experimentally determined values, or as upper or lower possible values if not measured directly), its biological or physical properties, and where applicable the absolute configuration and optical rotation.

The optical rotation in solution may be reported using the data name **_chemical_optical_rotation** by an expression of the form

$$[\alpha]_W^T = \pm\frac{100\alpha}{lc} \qquad (c = CONC, \mathrm{SOLV}),$$

where $[\alpha]_W^T$ is the signed optical rotation in degrees at temperature T and wavelength labelled by code W, l is the length of the optical cell, *CONC* is the concentration of the solution (given as the mass of the substance in g in a standard 100 ml of solution), and SOLV is the chemical formula of the solvent. This can be marked up within the constraints of the ASCII character set to which CIF is restricted as `[\a]^25^~D~ = +108 (c = 3.42, CHCl~3~)`, where the measurement is taken using the *D* line of the atomic spectrum of sodium.

Data items in the CHEMICAL_FORMULA category describe the chemical formula and formula mass of the compound under study. The quoted formula must reflect the overall stoichiometry of the crystal under study, and must, when multiplied by the Z value **_cell_formula_units_Z**, account for the total contents of the unit cell.

A number of data names are provided to account for different conventions in the presentation of chemical formulae. **_chemical_formula_analytical** is appropriate for a gross formula determined by standard chemical analysis, including all trace elements identified in the sample. Standard uncertainties on the proportions of elements present are acceptable, *e.g.*

```
_chemical_formula_analytical 'Fe2.45(2) Ni1.60(3) S4'
```

_chemical_formula_sum is another aggregate formula, in which all discrete bonded residues and ions are summed over the constituent elements. Where appropriate, the formulae of separate residues of a complex may be described by **_chemical_formula_moiety**, in which the formula for each moiety is supplied as a sum of the individual elements within the moiety, or by **_chemical_formula_structural**, in which sub-components within individual moieties are further identified, so that the overall expression permits the identification of particular bonded groups. Within these formula expressions, certain rules must be observed to allow parsing by software. The final data item relating to the chemical formula, **_chemical_formula_iupac**, is for formulae that are constructed according to the rules of the International Union for Pure and Applied Chemistry.

The ordering and notation rules are explained n detail in the dictionary, but are repeated here for convenience. Within each group of atoms for which a formula is present:

(i) only recognized element symbols may be used;

(ii) each element symbol is followed by a 'count' number ('1' is implicit and may be omitted);

(iii) a space or parenthesis must separate each cluster of (element symbol + count);

(iv) where a group of elements is enclosed in parentheses, the multiplier for the group must follow the closing parentheses. That is, all element and group multipliers are assumed to be printed as subscripted numbers. (An exception to this rule exists for **_chemical_formula_moiety**, where pre- and post-multipliers are permitted for molecular units.)

(v) Unless the elements are ordered in a manner that corresponds to their chemical structure, as in **_chemical_formula_structural**, the order of the elements within any group or moiety depends on whether or not carbon is present. If carbon is present, the order should be: C, then H, then the other elements in alphabetical order of their symbol. If carbon is not present, the elements are listed purely in alphabetic order of their symbol. This is the 'Hill' system used by *Chemical Abstracts*. This ordering is used in **_chemical_formula_moiety** and **_chemical_formula_sum**.

For **_chemical_formula_moiety** some additional rules apply:

(i) Moieties are separated by commas, ','.

(ii) The order of elements within a moiety follows the general rules outlined above as the 'Hill' system.

(iii) Parentheses are *not* used within moieties but may surround a moiety. Parentheses may not be nested.

(iv) Charges should be placed at the end of the moiety. The charge '+' or '-' may be preceded by a numerical multiplier and should be separated from the last (element symbol + count) by a space. Pre- or post-multipliers may be used for individual moieties.

Example 3.2.4.6 illustrates the differences between some of these data items.

3.2.4.2.2. *Chemical connectivity*

The data items in these categories are as follows:

(*a*) CHEMICAL_CONN_ATOM

```
• _chemical_conn_atom_number
  _chemical_conn_atom_charge
  _chemical_conn_atom_display_x
  _chemical_conn_atom_display_y
  _chemical_conn_atom_NCA
  _chemical_conn_atom_NH
  _chemical_conn_atom_type_symbol
```

(*b*) CHEMICAL_CONN_BOND

```
• _chemical_conn_bond_atom_1
        → _chemical_conn_atom_number
• _chemical_conn_bond_atom_2
        → _chemical_conn_atom_number
  _chemical_conn_bond_type
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

The CHEMICAL_CONN_ATOM category labels the chemical atoms in a connected representation of the molecular species and can also give the coordinates for the atoms in a two-dimensional chemical diagram (Example 3.2.4.7). Each atom may also carry an indication of the number of connected non-hydrogen atoms (**_NCA**) and the number of hydrogen atoms (**_NH**) to which it

Example 3.2.4.6. *Different representations of a chemical formula.*

```
_chemical_formula_iupac    '[Mo (C O)4 (C18 H33 P)2]'
_chemical_formula_moiety    'Mo,4(C O),2(C18 H33 P)'
_chemical_formula_structural
                          '((C O)4 (P (C6 H11)3)2)Mo'
_chemical_formula_sum              'C40 H66 Mo O4 P2'
```

Example 3.2.4.7. *Representation of a two-dimensional chemical diagram.*

```
loop_
     _chemical_conn_atom_number
     _chemical_conn_atom_type_symbol
     _chemical_conn_atom_display_x
     _chemical_conn_atom_display_y
     _chemical_conn_atom_NCA
     _chemical_conn_atom_NH
        1   S    .39  .81   1   0
        2   S    .39  .96   2   0
        3   N    .14  .88   3   0
        4   C    .33  .88   3   0
        5   C    .11  .96   2   2
        6   C    .03  .96   2   2
        7   C    .03  .80   2   2
        8   C    .11  .80   2   2
        9   S    .54  .81   1   0
        10  S    .54  .96   2   0
        11  N    .80  .88   3   0
        12  C    .60  .88   3   0
        13  C    .84  .96   2   2
        14  C    .91  .96   2   2
        15  C    .91  .80   2   2
        16  C    .84  .80   2   2
```

is connected. Together with the CHEMICAL_CONN_BOND category, the data items in the CHEMICAL_CONN_ATOM category provide a basic description of the chemical structure. Although the description of the chemical structure provided in these two categories is not as extensive as the information that may be conveyed in a molecular information file (Chapter 2.4), it should allow a substructure to be searched for in a suitable database.

The CHEMICAL_CONN_BOND category lists pairs of atoms that contribute to chemical bonds and describes the nature of the bond between them (Example 3.2.4.8). Taken with data items in the CHEMICAL_CONN_ATOM category, data items in this category complete the basic description of a molecular entity.

Bond types are assigned from a list that specifies single, double, triple, quadruple, aromatic, polymeric, delocalized double and π bonds. These are not intended to cover all possible cases, but to characterize a molecular model suitable for database substructure searching.

3.2.4.3. Molecular or packing geometry

The categories describing geometry are as follows:

GEOM group

- GEOM
- GEOM_ANGLE
- GEOM_BOND
- GEOM_CONTACT
- GEOM_HBOND
- GEOM_TORSION

The molecular and packing geometry can be calculated fully given the unit-cell parameters, the space group and a list of atom sites. Therefore, all the information about geometry in the GEOM category group is derivative. However, it is useful to record it within the file both as a check on the primary information stored in other categories and as a method for flagging values to be published.

Example 3.2.4.8. *Bond types in a chemical connectivity table.*

```
loop_
     _chemical_conn_bond_atom_1
     _chemical_conn_bond_atom_2
     _chemical_conn_bond_type
       4     1     doub     4     3     sing
       4     2     sing     5     3     sing
       6     5     sing     7     6     sing
       8     7     sing     8     3     sing
       10    2     sing     12    9     doub
       12    11    sing     12    10    sing
       13    11    sing     14    13    sing
       15    14    sing     16    15    sing
       16    11    sing     17    5     sing
       18    5     sing     19    6     sing
       20    6     sing     21    7     sing
       22    7     sing     23    8     sing
       24    8     sing     25    13    sing
       26    13    sing     27    14    sing
       28    14    sing     29    15    sing
       30    15    sing     31    16    sing
       32    16    sing
```

3.2.4.3.1. *Contents of the geometry-related categories*

The data items in these categories are as follows:

(*a*) GEOM

```
_geom_special_details
```

(*b*) GEOM_ANGLE

- _geom_angle_atom_site_label_1 → _atom_site_label
- _geom_angle_atom_site_label_2 → _atom_site_label
- _geom_angle_atom_site_label_3 → _atom_site_label
- _geom_angle_site_symmetry_1
- _geom_angle_site_symmetry_2
- _geom_angle_site_symmetry_3

```
_geom_angle
_geom_angle_publ_flag
```

(*c*) GEOM_BOND

- _geom_bond_atom_site_label_1 → _atom_site_label
- _geom_bond_atom_site_label_2 → _atom_site_label
- _geom_bond_site_symmetry_1
- _geom_bond_site_symmetry_2

```
_geom_bond_distance
_geom_bond_publ_flag
_geom_bond_valence
```

(*d*) GEOM_CONTACT

- _geom_contact_atom_site_label_1 → _atom_site_label
- _geom_contact_atom_site_label_2 → _atom_site_label
- _geom_contact_site_symmetry_1
- _geom_contact_site_symmetry_2

```
_geom_contact_distance
_geom_contact_publ_flag
```

(*e*) GEOM_HBOND

- _geom_hbond_atom_site_label_A → _atom_site_label
- _geom_hbond_atom_site_label_D → _atom_site_label
- _geom_hbond_atom_site_label_H → _atom_site_label
- _geom_hbond_site_symmetry_A
- _geom_hbond_site_symmetry_D
- _geom_hbond_site_symmetry_H

```
_geom_hbond_angle_DHA
_geom_hbond_distance_DA
_geom_hbond_distance_DH
_geom_hbond_distance_HA
_geom_hbond_publ_flag
```

(*f*) GEOM_TORSION

- _geom_torsion_atom_site_label_1
  → _atom_site_label
- _geom_torsion_atom_site_label_2
  → _atom_site_label
- _geom_torsion_atom_site_label_3
  → _atom_site_label
- _geom_torsion_atom_site_label_4
  → _atom_site_label
- _geom_torsion_site_symmetry_1
- _geom_torsion_site_symmetry_2
- _geom_torsion_site_symmetry_3
- _geom_torsion_site_symmetry_4

  _geom_torsion
  _geom_torsion_publ_flag

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. ***_symmetry_**** *items have a default value and may be omitted from the list. The arrow (→) is a reference to a parent data item.*

Most categories within this group record distances or angles specified by atom-site labels and are well characterized. The GEOM category currently provides the single data name **_geom_special_details** in which any other details of the geometry that an author considers noteworthy may be stored. Examples of information that might be stored in this data item are least-squares equations of planes, out-of-plane distances, dihedral angles between planes and general comments about the calculation of standard uncertainties.

A subtlety in the geometry-related categories arises from the need to record geometric relationships that involve atoms that are not listed in the ATOM_SITE coordinate list, but that can be derived from the coordinates in this list by the application of a crystallographic symmetry transformation. Thus atom sites in the geometry lists are identified both by their atom-site labels (which must identically match one of the entries in the ATOM_SITE list) *and* by the code for the symmetry transformation that has been applied to the initial location. Since the atom-site labels may refer to atoms in their original location as well as to atoms in symmetry-related locations, the formal key for these categories involves the site labels as well as the symmetry codes. However, in many cases (as discussed further below) the symmetry codes may be absent from a list, and a parser must supply suitable default or null values for the missing components when constructing or checking a complete key.

In many cases, interest is focused on intramolecular distances and angles, and on intramolecular contacts within a single asymmetric unit. In such cases, the geometry lists would contain only atoms listed explicitly in the ATOM_SITE list and the symmetry codes all refer trivially to the identity transformation.

The examples in this section demonstrate various ways of handling geometry lists with trivial or non-trivial symmetry transformations. In Example 3.2.4.9, showing treatment of bond angles, the relevant data items (**_geom_angle_site_symmetry_***) are absent, which is one method for indicating the identity transformation. Dictionary validation software must therefore be able to handle both the presence and absence of these components of the formal category key.

The symmetry transformations in this and related categories take the form of codes *'n klm'* or *n_klm*, where *n* refers to the symmetry operation that is applied to the coordinates stored in **_atom_site_fract_x**, **_atom_site_fract_y** and **_atom_site_fract_z**. The value of *n* must match a number given in **_symmetry_equiv_pos_site_id**. *k*, *l* and *m* refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the contact. These translations (x, y, z) are related to (k, l, m) by

Example 3.2.4.9. *List of bond angles.*

```
loop_
     _geom_angle_atom_site_label_1
     _geom_angle_atom_site_label_2
     _geom_angle_atom_site_label_3
     _geom_angle
     _geom_angle_publ_flag
     C2  O1  C5   111.6(2)  yes
     O1  C2  C3   110.9(2)  yes
     O1  C2  O21  122.2(3)  yes
     C3  C2  O21  127.0(3)  yes
     C2  C3  N4   101.3(2)  yes
     C2  C3  C31  111.3(2)  yes
     C2  C3  H3   107(1)    no
     N4  C3  C31  116.7(2)  yes
```

Example 3.2.4.10. *List of bonds.*

```
loop_
     _geom_bond_atom_site_label_1
     _geom_bond_atom_site_label_2
     _geom_bond_distance
     _geom_bond_site_symmetry_1
     _geom_bond_site_symmetry_2
     _geom_bond_publ_flag
      O1  C2   1.342(4)  1_555  1_555  yes
      O1  C5   1.439(3)  1_555  1_555  yes
      C2  C3   1.512(4)  1_555  1_555  yes
      C2  O21  1.199(4)  1_555  1_555  yes
      C3  N4   1.465(3)  1_555  1_555  yes
      C3  C31  1.537(4)  1_555  1_555  yes
      C3  H3   1.00(3)   1_555  1_555  no
      N4  C5   1.472(3)  1_555  1_555  yes
```

$$k = 5 + x, \qquad l = 5 + y, \qquad m = 5 + z.$$

By adding 5 to the translations, the use of negative numbers is avoided. As an example, the symmetry code 7_645 means that the symmetry operation with label '7' in the **_symmetry_equiv_pos_site_id** list is applied and the resulting position is translated $+1.0 \times a$ along the x axis, $-1.0 \times b$ along the y axis and $0.0 \times c$ along the z axis, where a, b and c are the unit-cell edges.

List entries with a **_geom_angle_publ_flag** value of `yes` are those that should be published.

The GEOM_BOND category records intramolecular bond distances. In Example 3.2.4.10, all the atoms are untransformed and are at the positions given in the ATOM_SITE list. The symmetry code is 1_555, where the trivial symmetry operation x, y, z is numbered '1' by **_symmetry_equiv_pos_site_id**.

The GEOM_CONTACT category records nonbonded interatomic contacts. In Example 3.2.4.11, all the atoms are untransformed and are at the positions given in the ATOM_SITE list, and therefore the symmetry codes all have the value '.' (meaning 'inapplicable'). This is another method for indicating the identity transformation.

The GEOM_HBOND category records details about hydrogen bonds. Unlike other categories in the GEOM group, the GEOM_HBOND category records information about both distances and angles, including donor–acceptor, donor–hydrogen and acceptor–hydrogen distances and the included angle at the hydrogen-atom site (see Example 3.2.4.12). The comments above about the interpretation of symmetry codes and their relevance in the formal assignment of the category key also apply to this category.

Note that, strictly speaking, this category should only be populated if coordinates for the hydrogen atom are available (because the mandatory component of the category key **_geom_hbond_**

Example 3.2.4.11. *List of nonbonded interatomic contacts.*

```
loop_
     _geom_contact_atom_site_label_1
     _geom_contact_atom_site_label_2
     _geom_contact_distance
     _geom_contact_site_symmetry_1
     _geom_contact_site_symmetry_2
     _geom_contact_publ_flag
     O(1)  O(2)     2.735(3)  .  .  yes
     H(O1) O(2)     1.82      .  .  no
```

Example 3.2.4.12. *List of hydrogen-bond distances and angles.*

```
loop_
     _geom_hbond_atom_site_label_D
     _geom_hbond_atom_site_label_H
     _geom_hbond_atom_site_label_A
     _geom_hbond_distance_DH
     _geom_hbond_distance_HA
     _geom_hbond_distance_DA
     _geom_hbond_angle_DHA
     _geom_hbond_publ_flag
N6  HN6 OW  0.888(8) 1.921(12) 2.801(8) 169.6(8) yes
OW  HO2 O7  0.917(6) 1.923(12) 2.793(8) 153.5(8) yes
OW  HO1 N10 0.894(8) 1.886(11) 2.842(8) 179.7(9) yes
```

Example 3.2.4.13. *List of torsion angles.*

```
    loop_
     _geom_torsion_atom_site_label_1
     _geom_torsion_atom_site_label_2
     _geom_torsion_atom_site_label_3
     _geom_torsion_atom_site_label_4
     _geom_torsion
     _geom_torsion_site_symmetry_1
     _geom_torsion_site_symmetry_2
     _geom_torsion_site_symmetry_3
     _geom_torsion_site_symmetry_4
     _geom_torsion_publ_flag
C(9)  O(2) C(7)  C(2)    71.8(2)  .  .  .  .      yes
C(7)  O(2) C(9)  C(10) -168.0(3)  .  .  .  2_666  yes
C(10) O(3) C(8)  C(6)  -167.7(3)  .  .  .  .      yes
C(8)  O(3) C(10) C(9)   -69.7(2)  .  .  .  2_666  yes
O(1)  C(1) C(2)  C(3)  -179.5(4)  .  .  .  .      no
O(1)  C(1) C(2)  C(7)    -0.6(1)  .  .  .  .      no
```

`atom_site_label_H` needs a parent label in the atom-site list). In practice, hydrogen bonds can be assumed between donor atoms and acceptors even if the hydrogen atom is not specifically located.

The items in the GEOM_TORSION category describe the torsion angle in degrees generated for the bonded sequence of four atom sites identified by the `_geom_torsion_atom_site_label_*` codes. As with other geometry-specific site labels, these must match labels specified as `_atom_site_label` in the atom list. The torsion angle definition is that of Klyne & Prelog (1960).

Example 3.2.4.13 includes two sites that have been generated by crystallographic symmetry operations and lattice translations from the parent sites in the atom list.

3.2.4.4. Symmetry and space-group information

The categories describing symmetry are as follows:

SYMMETRY group

Original symmetry categories (§3.2.4.4.1)

SYMMETRY

SYMMETRY_EQUIV

Replacement symmetry categories (§3.2.4.4.2)

SPACE_GROUP

SPACE_GROUP_SYMOP

The SPACE_GROUP and older SYMMETRY categories contain information about the symmetry of the crystal; specifically the space group and the symmetry-equivalent positions for that space group. More information about the symmetry is available in the symCIF dictionary described in Chapter 3.8 and presented in Chapter 4.7. The categories SPACE_GROUP and SPACE_GROUP_SYMOP were imported from symCIF in version 2.3 of the core dictionary, and are intended to replace the SYMMETRY and SYMMETRY_EQUIV categories. In most cases, there are strict equivalences between data items in the two sets. The new categories have been adopted for greater compatibility with future expansions to the symmetry CIF dictionary, and to correct some potentially misleading practices in the original categories. Although all the data items in SYMMETRY and SYMMETRY_EQUIV_POS are now formally marked as deprecated, it is likely that the older data items will remain in circulation for some time.

3.2.4.4.1. *Data items in SYMMETRY and related categories*

The data items in these categories are as follows:

(*a*) SYMMETRY

† `_symmetry_cell_setting`
† `_symmetry_Int_Tables_number`
† `_symmetry_space_group_name_H-M`
† `_symmetry_space_group_name_Hall`

(*b*) SYMMETRY_EQUIV

†• `_symmetry_equiv_pos_site_id`
† `_symmetry_equiv_pos_as_xyz`

The bullet (•) *indicates a category key. In practice* `_symmetry_equiv_pos_site_id` *is often absent from older CIFs. The dagger* (†) *indicates a deprecated item, which should not be used in the creation of new CIFs.*

The data items in the SYMMETRY category (now superseded by SPACE_GROUP) were used to record the space group. The Hermann–Mauguin (H-M) symbol was given by `_symmetry_space_group_name_H-M`. The dictionary definition recommended the use of the 'full' H-M symbol as listed in *International Tables for Crystallography* Volume A, but was not explicit about the meaning of 'full'. The dictionary examples showed short-form symbols expanded to a complete representation of individual symmetry elements; thus *Pnnn* would be given as `'P 2/n 2/n 2/n'`, and the monoclinic space group $P2_1/m$ would be given as `'P 1 21/m 1'` for the *b*-axis unique setting or `'P 1 1 21/m'` for the *c*-axis unique setting.

In practice, abbreviated symbols were often used, following conventions established over many years; thus `'P 21/m'` was often given as the Hermann–Mauguin symbol when the 'usual' *b* setting of a monoclinic cell had been chosen. It is recommended that these conventions should continue to be followed when the new data item `_space_group_name_H-M_alt` is used instead.

The dictionary examples also suggested concise ways of indicating the origin choice within the `_symmetry_space_group_name_H-M` field; since there is no formal description of how to do this, different authors used different wording. Hence, `_symmetry_space_group_name_H-M` was always best considered as a container for the representation of the space group that would appear in a published article, and not as a machine-readable source of information about the crystallographic symmetry.

The two mechanisms for conveying the symmetry transformations in a fully machine-readable form were the Hall symbol `_symmetry_space_group_name_Hall` (Hall, 1981; Hall & Grosse-Kunstleve, 2001) and a complete listing of the symmetry operations using data items in the SYMMETRY_EQUIV category.

The data item `_symmetry_cell_setting` indicates the crystal system, not (as suggested by its name) the setting used.

Example 3.2.4.14. *A list of symmetry-equivalent positions.*

```
loop_
     _symmetry_equiv_pos_site_id
     _symmetry_equiv_pos_as_xyz
        1     x,y,z
        2     1/2-x,-y,1/2+z
        3     1/2+x,1/2-y,-z
        4     -x,1/2+y,1/2-z
```

The SYMMETRY_EQUIV category, now superseded by SPACE_GROUP_SYMOP, provided a list of symmetry-equivalent positions in algebraic notation. Formally, **_symmetry_equiv_pos_site_id** acted as a category key, with any arbitrary numeric value that uniquely identifies each operator. Historically, the earliest versions of the core dictionary did not have such an identifier at all and the separate equivalent positions were indexed by their position in the **_symmetry_equiv_pos_as_xyz** list. This interpretation was vulnerable to inadvertent re-ordering of the list of equivalent positions, and for this reason, as well as to satisfy the formal need for a category key, **_symmetry_equiv_pos_site_id** was added (Example 3.2.4.14). For compatibility with software that was written to handle the earlier arrangement, it is recommended that **_symmetry_equiv_pos_site_id** gives sequential integer labels, starting with 1, to the equivalent positions in the sequence in which they appear in the CIF.

Note that the **_symmetry_equiv_pos_as_xyz** list must contain *all* symmetry-equivalent positions of the space group, including those generated by lattice centring and a centre of symmetry, if present.

3.2.4.4.2. *Data items in SPACE_GROUP and related categories*

Data items in these categories are as follows:

(*a*) SPACE_GROUP

_space_group_crystal_system
_space_group_id
_space_group_IT_number
_space_group_name_Hall
_space_group_name_H-M_alt

(*b*) SPACE_GROUP_SYMOP

• **_space_group_symop_id**
_space_group_symop_operation_xyz
_space_group_symop_sg_id

The bullet (•) *indicates a category key.*

The data items in the SPACE_GROUP category record the space group and crystal system. They recognize the common practice of supplying the space group in Hermann–Mauguin notation, though the H-M symbol does not contain complete information about the symmetry and the space-group origin. **_space_group_name_H-M_alt** allows the use of any legitimate H-M symbol as listed in *International Tables for Crystallography* Volume A or derived by similar principles. It does not give rigorous direction on how the symbols should be presented. It is recommended that the use of this symbol in CIFs containing articles for publication should follow the guidelines for **_symmetry_space_group_name_H-M** (Section 3.2.4.4.1).

Because a given space-group type may be described by more than one Hermann–Mauguin symbol, the space-group type should be specified by the use of **_space_group_IT_number**.

Two mechanisms exist for conveying fully machine-readable descriptions of the symmetry transformations relevant to the space group and setting. The first is the Hall symbol (Hall, 1981; Hall & Grosse-Kunstleve, 2001), which uniquely defines the space group and its reference to a particular coordinate system; it is specified in the data item **_space_group_name_Hall**. Alternatively, the symmetry operations may be listed in full using data items in the SYMMETRY_EQUIV category.

Example 3.2.4.15. *A list of symmetry operators using data items from the SPACE_GROUP_SYMOP category.*

```
loop_
     _space_group_symop_id
     _space_group_symop_operation_xyz
        1     x,y,z
        2     1/2-x,-y,1/2+z
        3     1/2+x,1/2-y,-z
        4     -x,1/2+y,1/2-z
```

The SPACE_GROUP_SYMOP category provides a list of the symmetry operators for a space group in algebraic notation. It replaces the category SYMMETRY_EQUIV_POS. Unlike the older category, where in practice the category key could be omitted from listings (and must therefore be generated implicitly by parsing software), the category key **_space_group_symop_id** *must* be given. See Example 3.2.4.15, which may be compared with Example 3.2.4.14.

3.2.4.5. Bond-valence information

Categories describing bond valences are as follows:

VALENCE group
  VALENCE_PARAM
  VALENCE_REF

Data items in these categories are as follows:

(*a*) VALENCE_PARAM

_valence_param_atom_1
_valence_param_atom_1_valence
_valence_param_atom_2
_valence_param_atom_2_valence
_valence_param_B
_valence_param_details
_valence_param_id
_valence_param_ref_id
  → **_valence_ref_id**
_valence_param_Ro

(*b*) VALENCE_REF

_valence_ref_id
_valence_ref_reference

The arrow (→) *is a reference to a parent data item.*

The data items in this category group relate to bond valences, which are widely used in inorganic crystallography to confirm and analyse the results of crystal structure determinations. Bond valences are determined from the bond lengths and have the useful property that their sum around any atom is equal to the atom valence (formal charge). They are increasingly being published with bond lengths. The data item **_geom_bond_valence** in the GEOM_BOND category allows the bond valence to be associated with the bond length.

The two categories discussed here list the parameters used to calculate the bond valences and their literature sources. These items might also be published, particularly where there is some uncertainty about the appropriate parameters to use.

The data items in the VALENCE_PARAM category define the parameters used for calculating bond valences from bond lengths. In addition to the parameters, a pointer to the reference for the source of the parameters (in VALENCE_REF) is given (Example 3.2.4.16).

Example 3.2.4.16. *A list of bond-valence parameters.*

```
loop_
     _valence_param_atom_1
     _valence_param_atom_1_valence
     _valence_param_atom_2
     _valence_param_atom_2_valence
     _valence_param_Ro
     _valence_param_B
     _valence_param_ref_id
     _valence_param_details
      Cu 2 O -2 1.679 0.37 a .
      Cu 2 O -2 1.649 0.37 j .
      Cu 2 N -3 1.64  0.37 m '2-coordinate N'
      Cu 2 N -3 1.76  0.37 m '3-coordinate N'
loop_
     _valence_ref_id
     _valence_ref_reference
a
'Brown & Altermatt (1985), Acta Cryst. B41, 244-247'
j
'Liu & Thorp (1993), Inorg. Chem. 32, 4102-4205'
m
; See, Krause & Strub (1998), Inorg. Chem.
  37, 5369-5375'
;
```

3.2.5. Publication

As an archival file format, CIF is well suited to the complete documentation of a structural study and the categories described in this section provide data items suitable for the generation of a fully documented report, either as an informal laboratory notebook document or as a formal published article.

3.2.5.1. Literature citations

The categories describing literature citations are as follows:

CITATION group

CITATION

CITATION_AUTHOR

CITATION_EDITOR

The entries in the CITATION category group provide a set of data items suitable for the structured recording of references to the literature. At present, they are designed for the storage and retrieval of information about journal articles and individual chapters in books. They do not currently cover conference proceedings, pamphlets, preprints, theses or other kinds of publication. Reference lists are usually requested by journals that accept articles in CIF format as a single text field in `_publ_section_references`, but the categories in the CITATION group may become more useful for storing citation lists in the future, especially if converters become available to and from other bibliographic formats such as EndNote and BibTEX.

Data items in these categories are as follows:

(*a*) CITATION

- `_citation_id`

`_citation_abstract`
`_citation_abstract_id_CAS`
`_citation_book_id_ISBN`
`_citation_book_publisher`
`_citation_book_publisher_city`
`_citation_book_title`
`_citation_coordinate_linkage`
`_citation_country`
`_citation_database_id_CSD`
`_citation_database_id_Medline`
`_citation_journal_abbrev`
`_citation_journal_full`
`_citation_journal_id_ASTM`
`_citation_journal_id_CSD`
`_citation_journal_id_ISSN`
`_citation_journal_issue`
`_citation_journal_volume`
`_citation_language`
`_citation_page_first`
`_citation_page_last`
`_citation_special_details`
`_citation_title`
`_citation_year`

(*b*) CITATION_AUTHOR

- `_citation_author_citation_id` → `_citation_id`

`_citation_author_name`
`_citation_author_ordinal`

(*c*) CITATION_EDITOR

- `_citation_editor_citation_id` → `_citation_id`

`_citation_editor_name`
`_citation_editor_ordinal`

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

The CITATION category provides the bulk of the information about individual citations. `_citation_id` provides a link to the CITATION_AUTHOR and CITATION_EDITOR categories, where multiple authors, and, if appropriate, multiple editors are listed.

Example 3.2.5.1 shows how a fully populated citation list is structured across these categories.

The authors of a cited reference are listed using items from the CITATION_AUTHOR category. The value of `_citation_author_citation_id` must match a value of `_citation_id` in the CITATION category, and this data item forms the link between the authors and the citations. `_citation_author_ordinal` is used to record the order in which the authors are listed.

The editors of a cited reference are listed using items from the CITATION_EDITOR category. The value of `_citation_editor_citation_id` must match a value of `_citation_id` in the CITATION category, and this data item forms the link between the editors and the citations. `_citation_editor_ordinal` is used to record the order in which the editors are listed.

Example 3.2.5.1. *A structured bibliographic reference list.*

```
loop_
     _citation_id
     _citation_title
     _citation_page_first
     _citation_page_last
     _citation_year
     _citation_journal_abbrev
     _citation_journal_volume
     _citation_journal_id_ISSN
      1
;     Angle calculations for 3- and 4-circle X-ray
      and neutron diffractometers
;
      457  464  1967  'Acta Cryst.'  22  0365-110X
      2
     'Space-group notation with an explicit origin'
      517  525  1981  'Acta Cryst. Section A'  37
      0108-7673
      3 ? 521  523  1960  'Experientia' 16 ?

loop_
     _citation_author_citation_id
     _citation_author_name
      1   'Busing, W. R.'
      1   'Levy, H. A.'
      2   'Hall, S. R.'
      3   'Klyne, W.'
      3   'Prelog, V.'
```

3.2.5.2. Citation of software packages

The single category describing software citations is as follows:

COMPUTING group
  COMPUTING

Data items in this category are as follows:

COMPUTING
  _computing_cell_refinement
  _computing_data_collection
  _computing_data_reduction
  _computing_molecular_graphics
  _computing_publication_material
  _computing_structure_refinement
  _computing_structure_solution

The items in this category identify the software packages used for particular stages in a standard small-molecule crystallographic study. They may of course be used in other types of study as long as the description implied by the data name is relevant. The mmCIF dictionary provides a more general category, SOFTWARE, for the structured recording of programs used for a wider range of purposes.

3.2.5.3. Citation of related database entries

The single category describing related database entries is as follows:

DATABASE group
  DATABASE

Data items in this category are as follows:

DATABASE
  _database_code_CAS
  _database_code_CSD
  _database_code_ICSD
  _database_code_MDF
  _database_code_NBS
  _database_code_PDB
  _database_code_PDF
  _database_code_depnum_ccdc_archive
  _database_code_depnum_ccdc_fiz
  _database_code_depnum_ccdc_journal
  _database_CSD_history
  _database_journal_ASTM
  _database_journal_CSD

The _database_code_ items store the identifiers provided by specific databases for the structure described in the current data block. In the order given above, the databases they refer to are: *Chemical Abstracts*, the Cambridge Structural Database, the Inorganic Crystal Structure Database, the Metals Data File, the Crystal Data database of the National Institute of Standards and Technology (formerly the National Bureau of Standards), the Protein Data Bank, and the Powder Diffraction File of the International Centre for Diffraction Data.

The _database_code_depnum_ccdc_* items record deposition numbers assigned to files containing structural information archived by the Cambridge Crystallographic Data Centre (CCDC). The deposition numbers are as assigned by the CCDC itself (*_archive), by the Fachinformationszentrum Karlsruhe (*_fiz) or by a journal (*_journal). The item _database_CSD_history records the history of changes made by the CCDC and incorporated into the Cambridge Structural Database.

The _database_journal_ items store, respectively, the coden designator for journal titles of the American Society for Testing and Materials (ASTM), as given in the *Chemical Source List* maintained by the *Chemical Abstracts* Service, and the journal code used in the Cambridge Structural Database.

These specific items are regarded as appropriate for small-molecule and inorganic structures. The mmCIF dictionary includes a DATABASE_2 category, where an extensible data scheme allows additional database entries to be stored without requiring a separate data item for each new database reference.

3.2.5.4. Journal housekeeping, citation and indexing entries

The categories used for journal housekeeping and indexing are as follows:

JOURNAL group
  JOURNAL
  JOURNAL_INDEX

The data items in the JOURNAL category group are concerned with the processing of an article for publication. They are used mainly by the staff of the editorial office of an academic journal and are of limited interest to the practising crystallographer. They are not defined explicitly in the core dictionary and are included here only for the sake of completeness.

(*a*) JOURNAL
  _journal_coden_ASTM
  _journal_coden_Cambridge
  _journal_coeditor_address
  _journal_coeditor_code
  _journal_coeditor_email
  _journal_coeditor_fax
  _journal_coeditor_name
  _journal_coeditor_notes
  _journal_coeditor_phone
  _journal_data_validation_number
  _journal_date_accepted
  _journal_date_from_coeditor
  _journal_date_to_coeditor
  _journal_date_printers_final
  _journal_date_printers_first
  _journal_date_proofs_in
  _journal_date_proofs_out
  _journal_date_recd_copyright
  _journal_date_recd_electronic
  _journal_date_recd_hard_copy
  _journal_issue
  _journal_language
  _journal_name_full
  _journal_page_first
  _journal_page_last
  _journal_paper_category
  _journal_suppl_publ_number
  _journal_suppl_publ_pages
  _journal_techeditor_address
  _journal_techeditor_code
  _journal_techeditor_email
  _journal_techeditor_fax
  _journal_techeditor_name
  _journal_techeditor_notes
  _journal_techeditor_phone
  _journal_volume
  _journal_year

(*b*) JOURNAL_INDEX
  _journal_index_subterm
  _journal_index_term
  _journal_index_type

Of the data items in the JOURNAL category, the only ones that are likely to be of interest to users other than the journal staff are the items recording the bibliographic information upon publication, namely _journal_name_full, _journal_year, _journal_volume, _journal_page_first and _journal_page_last.

Data items in the JOURNAL_INDEX category allow terms to be embedded within a CIF that will be used for generating journal indexes. Example 3.2.5.2 shows how this is done; the possible values of _journal_index_type are defined by the journal and for *Acta Crystallographica* and other IUCr journals may be one of S (subject index), I (inorganic formula index), M (metal-organic formula index) or O (organic formula index).

Example 3.2.5.2. *Markup of indexing terms.*

```
loop_
     _journal_index_type
     _journal_index_term
     _journal_index_subterm
      O   C16H19NO4             .
      S   alkaloids            (-)-norcocaine
      S   (-)-norcocaine        .
      S
; [2R,3S-(2,3)]-methyl
  3-(benzoyloxy)-8-azabicyclo[3.2.1]octane-2-
  carboxylate
;                          .
```

Example 3.2.5.3. *Request to add material for publication to a journal's standard list.*

```
loop_
     _publ_manuscript_incl_extra_item
     _publ_manuscript_incl_extra_info
'_atom_site_symmetry_multiplicity'
     'to emphasise special sites'
'_chemical_compound_source'
     'rare material, unusual source'
'_reflns_d_resolution_high'
     'limited data are a problem here'
```

3.2.5.5. Contents of a publication

Categories used to describe an article for publication and to include the text of an article are as follows:

PUBL group

- PUBL
- PUBL_AUTHOR
- PUBL_BODY
- PUBL_MANUSCRIPT_INCL

The items in the PUBL category group describe the text that an author adds to the experimental data in a CIF to create a full record of the structural study for publication.

Data items in these categories are as follows:

(*a*) PUBL

```
_publ_contact_author
_publ_contact_author_address
_publ_contact_author_email
_publ_contact_author_fax
_publ_contact_author_id_iucr
_publ_contact_author_name
_publ_contact_author_phone
_publ_contact_letter
_publ_manuscript_creation
_publ_manuscript_processed
_publ_manuscript_text
_publ_requested_category
_publ_requested_coeditor_name
_publ_requested_journal
_publ_section_abstract
_publ_section_acknowledgements
_publ_section_comment
_publ_section_discussion
_publ_section_experimental
_publ_section_exptl_prep
_publ_section_exptl_refinement
_publ_section_exptl_solution
_publ_section_figure_captions
_publ_section_introduction
_publ_section_references
_publ_section_synopsis
_publ_section_table_legends
_publ_section_title
_publ_section_title_footnote
```

(*b*) PUBL_AUTHOR

```
_publ_author_address
_publ_author_email
_publ_author_footnote
_publ_author_id_iucr
_publ_author_name
```

(*c*) PUBL_BODY

```
_publ_body_contents
_publ_body_element
_publ_body_format
_publ_body_label
_publ_body_title
```

(*d*) PUBL_MANUSCRIPT_INCL

```
_publ_manuscript_incl_extra_defn
_publ_manuscript_incl_extra_info
_publ_manuscript_incl_extra_item
```

The data items in the PUBL category represent non-looped components of the published article, varying from the article title to the complete text of the article. Some journals such as *Acta Crystallographica* require specific section headers in articles, for which data items (*e.g.* `_publ_section_comment`) are provided. An alternative approach is to use the general items in this list for the article title, abstract, reference list *etc.* and build the individual sections of text using the items in the PUBL_BODY category.

The CIF syntax restrictions that permit only printable ASCII characters (Chapter 2.2) mean that authors cannot simply cut and paste text produced by commercial word-processing programs into a CIF. This might be inconvenient for the author, but while commercial word-processing programs are often convenient to use, they use proprietary and often poorly documented formats. For an archived CIF to remain readable in the long term, the use of transparent text representations, using open and well documented markup systems such as XML or TEX, is preferred.

The authors of an article are listed separately using items in the PUBL_AUTHOR category. The entry for each author can be annotated, for example to add text that would appear as a footnote to the author's name in the published article.

The PUBL_BODY category allows the body of an article to be more highly structured than `_publ_manuscript_text` does. It may be used for articles that include structural data but are less formally structured than required by *Acta Crystallographica Section C* or *Acta Crystallographica Section E*.

Journals like *Acta Crystallographica Section C* may have a list of CIF data items that will normally be published. If an author wishes to include additional data items, they can be specified using the _PUBL_MANUSCRIPT_INCL category. Since the *values* of `_publ_manuscript_incl_extra_item` are data names, they *must* be placed in quotes, as in Example 3.2.5.3, for them to be parsed correctly.

Further information on the use of the data items in the PUBL category group may be found in Section 5.7.2.

3.2.6. File metadata

The categories describing the history of a data block and its relation to other blocks are as follows:

AUDIT group

- AUDIT
- AUDIT_AUTHOR
- AUDIT_CONFORM
- AUDIT_CONTACT_AUTHOR
- AUDIT_LINK

Information about the origin and purpose of a CIF is needed to be able to make full use of the content of the CIF. Information about the CIF itself (rather than the experiment or structural model it describes) is known as *metadata*.

Because the scope of any data value is restricted to the data block in which it resides, each data block should contain its own set of **_audit_*** data items (a requirement that is often overlooked in the construction of a CIF with multiple data blocks). The data items in the AUDIT_LINK category may be used to record relationships between different data blocks within the same file.

Data items in these categories are as follows:

(*a*) AUDIT

```
_audit_block_code
_audit_creation_date
_audit_creation_method
_audit_update_record
```

(*b*) AUDIT_AUTHOR

```
_audit_author_address
_audit_author_name
```

(*c*) AUDIT_CONFORM

```
_audit_conform_dict_location
_audit_conform_dict_name
_audit_conform_dict_version
```

(*d*) AUDIT_CONTACT_AUTHOR

```
_audit_contact_author_address
_audit_contact_author_email
_audit_contact_author_fax
_audit_contact_author_name
_audit_contact_author_phone
```

(*e*) AUDIT_LINK

```
_audit_link_block_code
_audit_link_block_description
```

The AUDIT category provides a small set of data names suitable for identifying a data block and recording its creation date and subsequent modifications. Each data block in a CIF is introduced by a string of the form **data_*xxxx***, where the block code ***xxxx*** is an arbitrary string. CIF offers no guidelines for choosing a block code, and there are many cases where the same string has been chosen to label data blocks in different files. The **_audit_block_code** data item is meant to encourage authors to provide a unique label for a data block. Also, as a separate data item, **_audit_block_code** has the advantage that it can be interrogated using standard CIF query tools; this is not true of the block code.

The core dictionary does not specify a procedure for choosing a unique identifier for the data block, but other dictionaries do. The modulated structures dictionary recommends specific naming procedures (Section 3.4.4.4) and the power dictionary supplies alternative data items designed to generate globally unique identifiers (Section 3.3.7.1).

Some applications modify the block code in the **data_*xxxx*** string. The value of **_audit_block_code** may not be changed arbitrarily to suit the convenience of external applications.

In Example 3.2.6.1, the **_audit_block_code** assigned is different from the data-block code; the creation date is expressed in the CIF date format convention of *yyyy-mm-dd* and the revision record is generated by adding material to the **_audit_update_record** field. Each addition has been prefixed with the date and initialled by the person who made the change. It is good practice to maintain a full record of any changes of substance to the contents of the data block.

Data items in the AUDIT_AUTHOR category record details of the author or authors of the data block. Where there is more than a single author, the names and addresses are looped. The use of these data items parallels that of the items in the PUBL_AUTHOR category; the difference is that the latter are used specifically to record details of authors of an article for publication. The AUDIT_AUTHOR category refers to the creators of a CIF data block regardless of its intended purpose.

Example 3.2.6.1. *Items identifying a data block and recording its revision history.*

```
data_example

_audit_block_code                  xyzzy_2002-04-05
_audit_creation_date               2002-04-05
_audit_creation_method             'SHELXL97'

_audit_update_record
; 2002-04-09  discussion added                        BM
  2002-04-17  coeditor number XY1234 assigned        BM
  2002-04-18  revised comment after referee report BM
;
```

Example 3.2.6.2. *The CIF dictionaries to which the data block conforms.*

```
loop_
  _audit_conform_dict_name
  _audit_conform_dict_version
  _audit_conform_dict_location
    cif_core.dic      2.3.1   .
    cif_pd.dic        1.0.1   .
    cif_local_my.dic  1.0
              /usr/local/dics/my_local_dictionary
```

Data items in the AUDIT_CONFORM category describe the version of the dictionary or dictionaries that contain the definitions of the data names in the current data block. It is very helpful to provide this information, so that applications software can locate the original definitions and validate the contents of the current data block against them (Example 3.2.6.2). The dictionary identifier **_audit_conform_dict_name** is essential. The version is less important, as the dictionaries are revised in such a way as to try to retain compatibility between versions, but may occasionally be useful if changes of substance have crept in between versions. The location specified by **_audit_conform_dict_location** is useful only for local applications; in general the public register of CIF dictionaries should be used to locate dictionary files (see Section 3.1.8.3).

Data items in the AUDIT_CONTACT_AUTHOR category record details of the name and address of the author to be contacted concerning the contents of the data block. The use of these data items parallels that of the items in the PUBL_CONTACT_AUTHOR category; the difference is that the latter are used specifically to record details of the contact author of an article for publication. The AUDIT_CONTACT_AUTHOR category refers to the creator of a CIF data block regardless of its intended purpose.

The original purpose of a CIF, to record the data relevant to a single-crystal structure determination, was quickly extended to include the creation of an article reporting several crystal structures, as well as to powder CIFs recording information about multiple phases, modulated-structure CIFs describing superimposed structures and macromolecular CIFs recording results of multiple refinement cycles. A mechanism is required to differentiate the purpose of an individual data block and its relationship to other data blocks in the same file. This is provided by the AUDIT_LINK category. Example 3.2.6.3 shows how a CIF of an article for publication might show the relationships between the data blocks in the file. Note that the link references the value of **_audit_block_code** in the referenced data block, *not* the data-block header string itself (although in this example, and in Example 3.2.6.4, they have the same value).

Example 3.2.6.3. *List of linked data blocks in a CIF.*

```
data_global

_audit_block_code       global

loop_
  _audit_link_block_code
  _audit_link_block_description
    .         'text of paper with two structures'
    morA_(1)  'structure 1 of 2'
    morA_(2)  'structure 2 of 2'
```

Example 3.2.6.4. *Complementary list of linked data blocks in a secondary block.*

```
data_morA_(1)

_audit_block_code       morA_(1)

loop_
  _audit_link_block_code
  _audit_link_block_description
    global    'text of paper with two structures'
    .         'structure 1 of 2'
    morA_(2)  'structure 2 of 2'
```

For many applications, it is enough for a statement of the links between the data blocks in a CIF to be included once only in the file, normally in the initial data block. However, for completeness and to permit consistency checking, it is best if the other data blocks in the file have complementary declarations (Example 3.2.6.4).

Current practice as described in the core dictionary restricts this reporting of links between data blocks to the contents of a single file. In principle, if `_audit_block_code` were known to have globally unique values in each distinct data block, the mechanism could be extended to permit inter-file linkage.

Appendix 3.2.1
Category structure of the core CIF dictionary

Table A3.2.1.1 provides an overview of the structure of the core CIF dictionary by informal category group and categories.

Table A3.2.1.1. *Categories in the core CIF dictionary*

Numbers in parentheses refer to the section of this chapter in which each category is described in detail.

- ATOM group (§3.2.4.1)
  - ATOM_SITE (§3.2.4.1.1)
  - ATOM_SITES (§3.2.4.1.2)
  - ATOM_TYPE (§3.2.4.1.3)
- AUDIT group (§3.2.6)
  - AUDIT (§3.2.6(*a*))
  - AUDIT_AUTHOR (§3.2.6(*b*))
  - AUDIT_CONFORM (§3.2.6(*c*))
  - AUDIT_CONTACT_AUTHOR (§3.2.6(*d*))
  - AUDIT_LINK (§3.2.6(*e*))
- CELL group (§3.2.2.1)
  - CELL (§3.2.2.1(*a*))
  - CELL_MEASUREMENT_REFLN (§3.2.2.1(*b*))
- CHEMICAL group (§3.2.4.2)
  - CHEMICAL (§3.2.4.2.1(*a*))
  - CHEMICAL_CONN_ATOM (§3.2.4.2.2(*a*))
  - CHEMICAL_CONN_BOND (§3.2.4.2.2(*b*))
  - CHEMICAL_FORMULA (§3.2.4.2.1(*b*))
- CITATION group (§3.2.5.1)
  - CITATION (§3.2.5.1(*a*))
  - CITATION_AUTHOR (§3.2.5.1(*b*))
  - CITATION_EDITOR (§3.2.5.1(*c*))
- COMPUTING group (§3.2.5.2)
  - COMPUTING (§3.2.5.2)
- DATABASE group (§3.2.5.3)
  - DATABASE (§3.2.5.3)
- DIFFRN group (§3.2.2.2)
  - DIFFRN (§3.2.2.2.1)
  - DIFFRN_ATTENUATOR (§3.2.2.2.2(*a*))
  - DIFFRN_DETECTOR (§3.2.2.2.4)
  - DIFFRN_MEASUREMENT (§3.2.2.2.3(*a*))
  - DIFFRN_ORIENT_MATRIX (§3.2.2.2.3(*b*))
  - DIFFRN_ORIENT_REFLN (§3.2.2.2.3(*c*))
  - DIFFRN_RADIATION (§3.2.2.2.2(*b*))
  - DIFFRN_RADIATION_WAVELENGTH (§3.2.2.2.2(*c*))
  - DIFFRN_REFLN (§3.2.2.2.5(*a*))
  - DIFFRN_REFLNS (§3.2.2.2.5(*b*))
  - DIFFRN_REFLNS_CLASS (§3.2.2.2.5(*c*))
  - DIFFRN_SCALE_GROUP (§3.2.2.2.5(*d*))
  - DIFFRN_SOURCE (§3.2.2.2.2(*d*))
  - DIFFRN_STANDARD_REFLN (§3.2.2.2.5(*e*))
  - DIFFRN_STANDARDS (§3.2.2.2.5(*f*))
- EXPTL group (§3.2.2.3)
  - EXPTL (§3.2.2.3(*a*))
  - EXPTL_CRYSTAL (§3.2.2.3(*b*))
  - EXPTL_CRYSTAL_FACE (§3.2.2.3(*c*))
- GEOM group (§3.2.4.3)
  - GEOM (§3.2.4.3.1(*a*))
  - GEOM_ANGLE (§3.2.4.3.1(*b*))
  - GEOM_BOND (§3.2.4.3.1(*c*))
  - GEOM_CONTACT (§3.2.4.3.1(*d*))
  - GEOM_HBOND (§3.2.4.3.1(*e*))
  - GEOM_TORSION (§3.2.4.3.1(*f*))
- JOURNAL group (§3.2.5.4)
  - JOURNAL (§3.2.5.4(*a*))
  - JOURNAL_INDEX (§3.2.5.4(*b*))
- PUBL group (§3.2.5.5)
  - PUBL (§3.2.5.5(*a*))
  - PUBL_AUTHOR (§3.2.5.5(*b*))
  - PUBL_BODY (§3.2.5.5(*c*))
  - PUBL_MANUSCRIPT_INCL (§3.2.5.5(*d*))
- REFINE group (§3.2.3.1)
  - REFINE (§3.2.3.1(*a*))
  - REFINE_LS_CLASS (§3.2.3.1(*b*))
- REFLN group (§3.2.3.2)
  - REFLN (§3.2.3.2.1)
  - REFLNS (§3.2.3.2.2(*a*))
  - REFLNS_CLASS (§3.2.3.2.2(*b*))
  - REFLNS_SCALE (§3.2.3.2.2(*c*))
  - REFLNS_SHELL (§3.2.3.2.2(*d*))
- SYMMETRY group (§3.2.4.4)
  - SPACE_GROUP (§3.2.4.4.2(*a*))
  - SPACE_GROUP_SYMOP (§3.2.4.4.2(*b*))
  - SYMMETRY (§3.2.4.4.1(*a*))
  - SYMMETRY_EQUIV (§3.2.4.4.1(*b*))
- VALENCE group (§3.2.4.5)
  - VALENCE_PARAM (§3.2.4.5(*a*))
  - VALENCE_REF (§3.2.4.5(*b*))

References

Busing, W. R. & Levy, H. A. (1967). *Angle calculations for 3- and 4-circle X-ray and neutron diffractometers. Acta Cryst.* **22**, 457–464.

Hall, S. R. (1981). *Space-group notation with an explicit origin. Acta Cryst.* A**37**, 517–525; erratum (1981), A**37**, 921.

Hall, S. R. & Grosse-Kunstleve, R. W. (2001). *International tables for crystallography*, Vol. B, *Reciprocal space*, edited by U. Shmueli, 2nd ed., Appendix A1.4.2.3. Dordrecht: Kluwer Academic Publishers.

Klyne, W. & Prelog, V. (1960). *Description of steric relationships across single bonds. Experientia*, **16**, 521–523.

3.3. Classification and use of powder diffraction data

BY B. H. TOBY

3.3.1. Introduction

Powder diffraction is used for many purposes. One important use is the determination of crystallographic models (crystal structures), particularly when single crystals are not available. Other common uses of powder diffraction include the identification of unknown materials; determining the amounts of different crystalline phases in a mixture; studies of phase transitions; and measuring changes in lattice constants with composition, pressure or temperature. Residual stress measurements are also frequently made using powder diffraction. Measurements of the degree of preferred orientation of grains in a processed material (texture or pole-figure measurements) are also carried out using powder diffraction; these are useful as they can be used to relate the engineering properties of a material to its processing conditions.

A wide range of instrumentation is used for powder diffraction. Synchrotron, sealed-tube and rotating-anode X-ray sources and both spallation and reactor neutron sources are used. When monochromatic radiation is used, diffraction intensities are measured as a function of the detector setting angle, 2θ. When polychromatic radiation is used, however, energy-dispersive detectors at fixed angles are used. For a pulsed neutron source, the neutron time-of-flight (TOF) is measured to provide energy-dispersive detection. Energy-dispersive X-ray diffraction is not common, but is used for certain specialized measurements such as *in situ* high-pressure studies. Most laboratory powder diffractometers have a single discrete detector, but a few use position-sensitive detectors (PSDs). Multiple detectors or PSDs are common for specialist instruments, for example at large national user facilities.

Historically, X-ray sensitive film in a diffraction camera was used to measure the intensities. At present, diffractometers are much more common than cameras, but film is an inexpensive method of area detection and cameras continue to have some advantages over diffractometers. X-ray film, however, has a non-linear sensitivity and a limited dynamic range, so electronic area detectors and image plates are replacing film for many applications.

3.3.2. Dictionary design considerations

The many applications of powder diffraction and the wide range of different instruments used dictate the design of several aspects of the powder diffraction CIF (pdCIF) data dictionary. In contrast to single-crystal measurements, where observations are reduced to instrument-independent structure factors, in powder diffraction it is not possible to interpret the data without a detailed knowledge of the type of instrument used. Another difference is that while single-crystal measurements are rarely made for any purpose other than structure determination, powder diffraction data have many non-crystallographic applications (*e.g.* identifying the phases in a mixture of unknown composition). In a single-crystal diffraction experiment, one sample yields one crystal structure. In a powder diffraction experiment, however, it is common for the sample to be a mixture of phases. Analysis of the diffraction data may result in multiple crystal structure models. It is also common to use multiple data sets, for example to fit one crystallographic model to both X-ray and neutron data simultaneously.

The powder dictionary was developed with the following basic objectives in mind. A pdCIF:

(i) should record the data-collection experiment as completely as possible and document any data analysis;

(ii) should be appropriate for the exchange of unprocessed measurements, so that it can be used in national laboratories and other shared facilities;

(iii) may contain more than one data set and/or crystal structure through the use of multiple data blocks;

(iv) may accommodate references between data blocks; and

(v) should recognize and accommodate data from as many different types of instruments as possible.

The CIF syntax, as opposed to the STAR File syntax, is not well suited to handling large multi-dimensional data sets. For some two-dimensional image formats, this deficiency was addressed by the development of imgCIF (see Chapters 2.3 and 3.7). It is also true that the CIF syntax is not well suited to storing unprocessed powder-diffraction measurements from the many instruments that use area detectors (particularly for the case of the three-dimensional data structures needed for modern TOF instruments). Even in these cases, however, diffraction intensities are commonly reduced to simpler representations, such as might be input to a Rietveld refinement program. The pdCIF definitions are intended for use with these reduced data sets.

3.3.3. pdCIF dictionary sections

At the time when the powder CIF dictionary was created, it was recommended practice to give data items in other dictionaries names that distinguished them from items in the core CIF dictionary. This recommendation proved cumbersome and was subsequently withdrawn, but pdCIF data items are all named in a way that distinguishes them clearly from core CIF dictionary data items: data names in the pdCIF dictionary all begin with the string `_pd_` and, likewise, the data categories in the pdCIF dictionary all begin with PD_.

The data items in the pdCIF dictionary are named according to the type of information they contain, rather than the way they will be used in a data structure. For example, data items relating to observed data have names beginning with `_pd_meas_`, even though some data items beginning with `_pd_meas_` may be assigned to the same category as items in other parts of the dictionary. The formal category assignments (see Section 3.1.10.1) were dictated by the aim to make pdCIF as versatile and simple to use as possible. As an example, one might want to put both the observed and calculated intensities in a single table (loop), as would be the case for a Rietveld refinement. This requires that the calculated and observed intensity data items are assigned to the same category. Note that this does not require that these items always appear in a single loop; calculated and observed intensity values could appear in

Affiliation: BRIAN H. TOBY, NIST Center for Neutron Research, National Institute of Standards and Technology, Gaithersburg, Maryland 20899-8562, USA.

Table 3.3.3.1. *Category groups defined in the powder CIF dictionary*

The groups are listed in the order in which they are described in this chapter.

Section	Category group	Subject covered
(*a*) Experimental measurements		
3.3.4.1	PD_CHAR	Characterization of a sample
3.3.4.1	PD_PREP	Preparation of a sample
3.3.4.2	PD_SPEC	Specimen used in an experiment
3.3.4.3	PD_CALIB	Calibration information
3.3.4.3	PD_INSTR	The experimental instrument
3.3.4.4	PD_MEAS	Raw measurements and instrumental settings
(*b*) Analysis		
3.3.5.1	PD_PROC	Processed settings
3.3.5.2	PD_CALC	Simulated settings
3.3.5.3	PD_PEAK	Diffraction peak table
3.3.5.4	REFLN	Reflection assignments and intensities
(*c*) Atomicity, chemistry and structure		
3.3.6.1	PD_PHASE	Phases present
(*d*) File metadata		
3.3.7.1	PD_BLOCK	Relationships between data blocks
3.3.4.3, 3.3.4.4, 3.3.5.1, 3.3.5.2	PD_DATA	Measured and simulated intensities

separate loops, for example, if the increment between data points differs.

This need to contain diverse items in a common 'looped' list has led the pdCIF dictionary to use category names in a different way from the other CIF dictionaries, in which CIF data items are usually named according to their category. In the pdCIF dictionary, data items that might appear in the loop for diffraction intensities are assigned to the category PD_DATA. Only one data item is named using this category as prefix, **_pd_data_point_id**. Another departure from the convention used in other dictionaries is that several **_pd_refln_*** data names are assigned to the category REFLN so that these items may be included in a loop with **_refln_*** items defined in the core CIF dictionary.

Table 3.3.3.1 summarizes the category groups in the pdCIF dictionary; the individual categories are listed alphabetically in Appendix 3.3.1. The appendix also lists for each category the section of this chapter in which the category is described.

The order in which the categories are discussed follows the scheme of Table 3.1.10.1, so that the contents of the dictionary are summarized under the headings *Experimental measurements* (Section 3.3.4), *Analysis* (Section 3.3.5), *Atomicity, chemistry and structure* (Section 3.3.6) and *File metadata* (Section 3.3.7). The pdCIF dictionary does not contribute any new data items relevant to publication beyond those already in the core CIF dictionary.

The data items in each category are listed below. Category keys, if specified, are listed first and are marked by a bullet (•); the remaining data items in each category are listed alphabetically. Note that the category PD_DATA is discussed in several different sections.

3.3.4. Experimental measurements

The categories in the powder CIF dictionary relating to the crystallographic experiment are as follows:

Characterization and preparation of the sample (§3.3.4.1)
PD_CHAR group
  PD_CHAR
PD_PREP group
  PD_PREP

Description of the specimen (§3.3.4.2)
PD_SPEC group
  PD_SPEC

Instrument calibration and design (§3.3.4.3)
PD_CALIB group
  PD_CALIB
  PD_CALIBRATION
PD_INSTR group
  PD_INSTR

Observations and measurement conditions (§3.3.4.4)
PD_DATA group
  PD_DATA (items beginning with **_pd_meas_***)
PD_MEAS group
  PD_MEAS_INFO
  PD_MEAS_METHOD

The pdCIF dictionary differentiates between the terms *sample* and *specimen*. The terms are often treated as interchangeable, but they have quite distinct meanings. The term sample refers to a batch of material, while the term specimen refers to the particular portion of the sample that was used for a measurement. In some cases, the specimen is modified before it is used for data collection. For example, it may be mixed with an internal standard, dried, hydrated or pressed into a pellet.

3.3.4.1. Characterization and preparation of the sample

The data items in these categories are as follows:

(*a*) PD_CHAR
```
_pd_char_atten_coef_mu_calc
_pd_char_atten_coef_mu_obs
_pd_char_colour
_pd_char_particle_morphology
_pd_char_special_details
```

(*b*) PD_PREP
```
_pd_prep_conditions
_pd_prep_cool_rate
_pd_prep_pressure
_pd_prep_temperature
```

The PD_CHAR data items describe information known about the sample from observation and chemical analysis. For example, a description of the sample morphology can be specified using **_pd_char_particle_morphology**. Note that there are data items in the core dictionary that are appropriate for use with powder diffraction. For example, **_atom_type_analytical_mass_%** can be used for chemical analysis results and **_chemical_melting_point** for the melting point. Several similar data items occur in the pdCIF and core dictionaries. **_exptl_crystal_colour** and **_pd_char_colour** both describe the sample colour, but **_pd_char_colour** is more systematic. Also, **_pd_char_atten_coef_mu_calc** and **_exptl_absorpt_coefficient_mu** describe similar properties, but **_pd_char_atten_coef_mu_calc** is adjusted for the sample packing fraction, so it can be compared with the experimental value, **_pd_char_atten_coef_mu_obs**, when a direct measurement is made.

The PD_PREP data items describe how the sample was collected or prepared. For example, **_pd_prep_pressure** and **_pd_prep_temperature** describe the pressure and temperature used to prepare the sample. Note that these will probably differ from the pressure and temperature conditions at which diffraction measurements are made. Measurement conditions are recorded in **_diffrn_ambient_pressure** and **_diffrn_ambient_temperature**.

3.3.4.2. Description of the specimen used in the experiment

The data items in this category are as follows:

PD_SPEC
- _pd_spec_description
- _pd_spec_mount_mode
- _pd_spec_mounting
- _pd_spec_orientation
- _pd_spec_preparation
- _pd_spec_shape
- _pd_spec_size_axial
- _pd_spec_size_equat
- _pd_spec_size_thick
- _pd_spec_special_details

The PD_SPEC data items describe the specimen used to measure the diffraction data. The data item _pd_spec_preparation describes how the specimen that was used to measure the diffraction data was treated, not how the sample was prepared (PD_PREP) or characterized (PD_CHAR).

The PD_SPEC data items are also used to describe how the specimen was mounted for the diffraction experiment. For example, _pd_spec_mount_mode and _pd_spec_orientation describe the measurement geometry, while _pd_spec_shape and _pd_spec_size_* describe the specimen shape and size.

3.3.4.3. Instrument calibration and description

The data items in these categories are as follows:

(*a*) PD_CALIB
- • _pd_calib_detector_id
- _pd_calib_2theta_offset
- _pd_calib_2theta_off_point
- _pd_calib_2theta_off_min
- _pd_calib_2theta_off_max
- _pd_calib_detector_response
- _pd_calib_std_external_block_id
- _pd_calib_std_external_name
- _pd_calib_std_internal_mass_%
- _pd_calib_std_internal_name

(*b*) PD_CALIBRATION
- _pd_calibration_conversion_eqn
- _pd_calibration_special_details

(*c*) PD_INSTR
- _pd_instr_2theta_monochr_pre
- _pd_instr_2theta_monochr_post
- _pd_instr_beam_size_ax
- _pd_instr_beam_size_eq
- _pd_instr_cons_illum_flag
- _pd_instr_cons_illum_len
- _pd_instr_dist_src/mono
- _pd_instr_dist_mono/spec
- _pd_instr_dist_src/spec
- _pd_instr_dist_spec/anal
- _pd_instr_dist_anal/detc
- _pd_instr_dist_spec/detc
- _pd_instr_divg_ax_src/mono
- _pd_instr_divg_ax_mono/spec
- _pd_instr_divg_ax_src/spec
- _pd_instr_divg_ax_spec/anal
- _pd_instr_divg_ax_anal/detc
- _pd_instr_divg_ax_spec/detc
- _pd_instr_divg_eq_src/mono
- _pd_instr_divg_eq_mono/spec
- _pd_instr_divg_eq_src/spec
- _pd_instr_divg_eq_spec/anal
- _pd_instr_divg_eq_anal/detc
- _pd_instr_divg_eq_spec/detc
- _pd_instr_geometry
- _pd_instr_location
- _pd_instr_monochr_pre_spec
- _pd_instr_monochr_post_spec
- _pd_instr_slit_ax_src/mono
- _pd_instr_slit_ax_mono/spec
- _pd_instr_slit_ax_src/spec
- _pd_instr_slit_ax_spec/anal
- _pd_instr_slit_ax_anal/detc
- _pd_instr_slit_ax_spec/detc
- _pd_instr_slit_eq_src/mono
- _pd_instr_slit_eq_mono/spec
- _pd_instr_slit_eq_src/spec
- _pd_instr_slit_eq_spec/anal
- _pd_instr_slit_eq_anal/detc
- _pd_instr_slit_eq_spec/detc
- _pd_instr_soller_ax_src/mono
- _pd_instr_soller_ax_mono/spec
- _pd_instr_soller_ax_src/spec
- _pd_instr_soller_ax_spec/anal
- _pd_instr_soller_ax_anal/detc
- _pd_instr_soller_ax_spec/detc
- _pd_instr_soller_eq_src/mono
- _pd_instr_soller_eq_mono/spec
- _pd_instr_soller_eq_src/spec
- _pd_instr_soller_eq_spec/anal
- _pd_instr_soller_eq_anal/detc
- _pd_instr_soller_eq_spec/detc
- _pd_instr_source_size_ax
- _pd_instr_source_size_eq
- _pd_instr_special_details

(*d*) Part of PD_DATA
- _pd_instr_var_illum_len

The bullet (•) *indicates a category key.*

Calibration information can be placed in the PD_CALIB and PD_CALIBRATION categories. The _pd_calibration_* data items are descriptive and will not appear in a loop. The _pd_calib_* items may be looped to describe multiple detectors. Correction values for 2θ can be given using the _pd_calib_2theta_offset and _pd_calib_2theta_off_* data items. A calibration equation can be given using _pd_calibration_conversion_eqn. When multiple detectors are used, _pd_calib_detector_response is used to indicate the relative performance of each detector. The detector deadtime is specified using the core data item _diffrn_detector_dtime (which cannot be looped by detector).

If an internal standard is added to the sample for calibration, this information is specified using _pd_calib_std_internal_name to specify the material added and _pd_calib_std_internal_mass_% to specify the amount.

When a set of calibration intensities is measured using an external standard, it is possible to include the measurements and the derived results in a separate CIF block. A data block would then use _pd_calib_std_external_block_id to link to the block containing the calibration information. See Section 3.3.7 for a discussion of block pointers and block IDs. Note that the use of a unique name for the block ID allows the calibration information to be stored in a separate file, so that the calibration CIF need not be repeated in every CIF that references it.

The PD_INSTR section of the pdCIF dictionary contains terms that describe the instrument used. For example, the instrument or laboratory location is given using _pd_instr_location. The instrument type can be indicated using _pd_instr_geometry. The instrument geometry can be described in much greater detail using several data items. The geometry is described in terms of four regions of the experiment: radiation source to monochromator (src/mono); monochromator to specimen (mono/spec); specimen to analyser (spec/anal); and analyser to detector (anal/detc). If no monochromator is present, the first two regions are combined into radiation source to specimen (src/spec). If no analyser is present, the last two regions are combined into specimen to detector (spec/detc). Thus two, three or four sets of values describe the dimensions of the instrument and the collimation. For example, _pd_instr_dist_src/mono would be used to specify the distance between the radiation source and the monochromator. Alternatively, _pd_instr_dist_src/spec would be used to specify the distance between the

radiation source and the specimen if no monochromator was present.

Two methods may be used to describe the slits limiting the divergence in the equatorial plane. The angular divergence allowed by the slits in degrees can be specified using the **_pd_instr_divg_eq_*** data items. Alternatively, the dimensions of the slits in the equatorial direction in millimetres may be specified using **_pd_instr_slit_eq_***. The dimensions of these slits in the axial direction, *i.e.* the direction perpendicular to the equatorial plane and containing the incident or diffracted beam as appropriate, are specified in millimetres using **_pd_instr_slit_ax_***. The axial slit lengths, along with the **_pd_instr_dist_*** distances, are useful for estimating the low-angle peak asymmetry (Finger *et al.*, 1994). Note that angular divergence in the axial plane is not a well defined concept for line-focus instruments, but can be specified, where appropriate, using **_pd_instr_divg_ax_***.

The axial and equatorial directions are shown schematically in Fig. 3.3.4.1. The equatorial plane contains the equatorial direction vectors, as well as the incident beam, the diffracted beam and the scattering vector. The axial plane is perpendicular to the equatorial plane and contains the sample centre, which is the point where the incident and diffracted beams meet. For area-detection instruments, the designations of axial and equatorial directions may be arbitrary.

Soller collimators are described using **_pd_instr_soller_eq_*** data items rather than **_pd_instr_divg_eq_*** data items. It is common practice to specify the Soller collimation in arc-minutes (*e.g.* 30′). However, pdCIF defines these items to have units of degrees, so 30′ would be recorded in the CIF as 0.5. It is not usual to limit the axial divergence, except to reduce low-angle asymmetry, but if this is done, the **_pd_instr_soller_ax_*** data items can be used to define this.

For constant-wavelength instruments, it is common to have a monochromator or filter either before the sample, or after the sample (an analyser), or sometimes both. This is described using **_pd_instr_monochr_pre_spec** and **_pd_instr_monochr_post_spec**. It is rare, but possible, to have both a filter and a monochromator in the same location. The BT-1 neutron powder diffractometer at NIST uses both a Cu(311) monochromator and a graphite filter to attenuate the $\lambda/2$ component. In this case, the two elements would be placed in a loop:

```
loop_  _pd_instr_monochr_pre_spec
  '5 seg. vert. focusing Cu(311) monochromator'
  '3 cm graphite filter'
```

Note that the monochromator and analyser takeoff angles are given using **_pd_instr_2theta_monochr_pre** and **_pd_instr_2theta_monochr_post**. It is useful to record these values for X-ray studies, as they are needed for proper polarization corrections.

In a conventional Bragg–Brentano diffractometer, the divergence slits limit the illumination area at the sample. However, since the φ axis (the sample θ axis) is usually set to bisect the 2θ angle of the detector, the actual length of the area of the sample that is illuminated changes with 2θ. One should choose divergence slits so that the beam does not illuminate areas outside the sample at the lowest diffraction angle used. An alternative method for data collection is to have a divergence slit that opens as 2θ increases, so that a constant area of the sample is illuminated. This is known as a θ-compensating slit. Using a θ-compensating slit provides a better signal-to-noise ratio at larger 2θ values, but means that the diffraction intensities have to be normalized to compensate for the change in illumination. It also introduces greater optical aberrations with 2θ; a flat plate becomes an increasingly worse approximation to the curved sample geometry in the true Bragg–Brentano geometry.

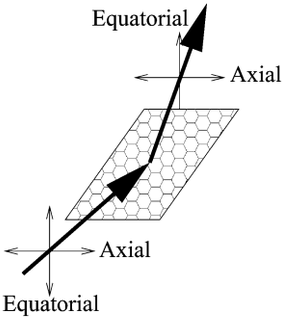


Fig. 3.3.4.1. The axial and equatorial directions in a powder-diffraction experiment.

The use of a variable divergence slit can be recorded in the form:

```
_pd_instr_cons_illum_flag      yes
_pd_instr_cons_illum_len       25.4
```

Note that if **_pd_instr_cons_illum_flag** is not specified, the value is assumed to be no, indicating that a fixed-width divergence slit has been used.

The beam size can be specified in two different ways: as the size at the source, using **_pd_instr_source_size_ax** and **_pd_instr_source_size_eq**, or as the size at the sample position, using **_pd_instr_beam_size_ax** and **_pd_instr_beam_size_eq**. Note that the size of the beam at the sample differs from the illumination length described above except when the sample is perpendicular to the beam. When a variable-divergence slit is in use, the beam size at the sample changes with 2θ, so if this size is known directly, the **_pd_instr_beam_size_eq** data item can be included in the loop containing the diffraction intensities. Similarly, in a constant-divergence instrument, where the illumination length changes with 2θ, the illumination length can be specified in the loop using **_pd_instr_var_illum_len**.

There are also several data items in the core CIF dictionary that should be present in the description of the instrument in a pdCIF. Use **_diffrn_radiation_probe** and **_diffrn_radiation_type** to specify the type of radiation used and **_diffrn_detector_type** to specify the detection type.

3.3.4.4. Observations and measurement conditions

The data items in these categories are as follows:

(*a*) Part of PD_DATA

```
_pd_data_point_id
_pd_meas_2theta_fixed
_pd_meas_2theta_range_min
_pd_meas_2theta_range_max
_pd_meas_2theta_range_inc
_pd_meas_2theta_scan
_pd_meas_angle_2theta
_pd_meas_angle_chi
_pd_meas_angle_omega
_pd_meas_angle_phi
_pd_meas_counts_total
_pd_meas_counts_background
_pd_meas_counts_container
_pd_meas_counts_monitor
_pd_meas_datetime_initiated
_pd_meas_detector_id
      → _pd_calib_detector_id
_pd_meas_point_id
_pd_meas_position
_pd_meas_rocking_angle
_pd_meas_step_count_time
_pd_meas_time_of_flight
```

(*b*) PD_MEAS_INFO
```
_pd_meas_info_author_address
_pd_meas_info_author_email
_pd_meas_info_author_fax
_pd_meas_info_author_name
_pd_meas_info_author_phone
_pd_meas_intensity_total
_pd_meas_intensity_background
_pd_meas_intensity_container
_pd_meas_intensity_monitor
```

(*c*) PD_MEAS_METHOD
```
_pd_meas_number_of_points
_pd_meas_rocking_axis
_pd_meas_scan_method
_pd_meas_special_details
_pd_meas_units_of_intensity
```

The arrow (→) is a reference to a parent data item.

The item `_pd_data_point_id` identifies each entry in the list of measured, processed or simulated intensities. It is the only item in the PD_DATA category that actually begins with the string `_pd_data_`, and is included here for convenience. If the list of intensities is split across several distinct loops, the role of this identifier may be adopted by other identifiers, such as `_pd_meas_point_id` in an isolated list of measured intensities.

The `_pd_meas_*` data items contain unprocessed measurements and documentation on the instrumental settings used for the measurements. Note that the choice of the data items used to represent this information is determined by the type of diffraction instrument, as well as how the measurement was conducted. This will be discussed further in Section 3.3.8. However, some `_pd_meas_*` data items are independent of the instrument type. For example, the use of `_pd_meas_datetime_initiated` is good practice, as is use of the `_pd_meas_info_author_*` data items. It is probably good practice to record the number of data points in `_pd_meas_number_of_points` for the benefit of people who might read the CIF, but there is no requirement that this item be present. This means that software should determine the number of data points directly when reading the CIF, rather than relying on the presence of a value for `_pd_meas_number_of_points`.

3.3.5. Analysis

The categories relating to the information derived from the measurements are as follows:

Processed intensities, positions and data processing (§3.3.5.1)
PD_DATA group
  PD_DATA (items beginning with `_pd_proc_*`)
PD_PROC group
  PD_PROC_INFO
  PD_PROC_LS
Simulated intensities and their positions (§3.3.5.2)
PD_DATA group
  PD_DATA (items beginning with `_pd_calc_*`)
PD_CALC group
  PD_CALC
Diffraction peak table (§3.3.5.3)
PD_PEAK group
  PD_PEAK
  PD_PEAK_METHOD
Reflection assignments and intensities (§3.3.5.4)
  REFLN

In Rietveld and other studies, processed or simulated intensities are presented alongside measured values. This leads to the presence of both derived and measured values in the same category (PD_DATA). However, the purposes of the data items that refer to processed and simulated data points are made clear by the way they are named. Overall descriptions of processed and simulated intensity data are covered by the categories PD_PROC_INFO, PD_PROC_LS and PD_CALC. The two categories PD_PEAK and PD_PEAK_METHOD are used to describe lists of peak positions, which would typically be used to search and match powder profiles. Some additional data items relevant to the table of Bragg reflections are defined as additions to the existing REFLN category in the core CIF dictionary.

3.3.5.1. Processed intensities, their positions and processing information

The data items in these categories are as follows:

(*a*) Part of PD_DATA
```
_pd_proc_2theta_corrected
_pd_proc_2theta_range_min
_pd_proc_2theta_range_max
_pd_proc_2theta_range_inc
_pd_proc_d_spacing
_pd_proc_energy_incident
_pd_proc_energy_detection
_pd_proc_intensity_net
_pd_proc_intensity_total
_pd_proc_intensity_bkg_calc
_pd_proc_intensity_bkg_fix
_pd_proc_intensity_incident
_pd_proc_intensity_norm
_pd_proc_point_id
_pd_proc_recip_len_Q
_pd_proc_wavelength
```

(*b*) PD_PROC_INFO
```
_pd_proc_info_author_address
_pd_proc_info_author_email
_pd_proc_info_author_fax
_pd_proc_info_author_name
_pd_proc_info_author_phone
_pd_proc_info_data_reduction
_pd_proc_info_datetime
_pd_proc_info_excluded_regions
_pd_proc_info_special_details
```

(*c*) PD_PROC_LS
```
_pd_proc_ls_background_function
_pd_proc_ls_peak_cutoff
_pd_proc_ls_pref_orient_corr
_pd_proc_ls_prof_R_factor
_pd_proc_ls_prof_wR_factor
_pd_proc_ls_prof_wR_expected
_pd_proc_ls_profile_function
_pd_proc_ls_special_details
_pd_proc_ls_weight
```

The pdCIF dictionary distinguishes between values that are measured directly and values that are derived from these observations. For example, in a constant-wavelength instrument, diffraction intensities are recorded as a function of 2θ. One may derive d-space values from the 2θ values using the value of the wavelength and corrections for the 2θ zero-point error and the sample displacement. One may also derive a new set of data points from the observations, for example by summing adjacent data points when the increment between the data points is much smaller than is warranted by the peak widths. For peak searching and other non-quantitative purposes, the diffraction intensities may be smoothed or otherwise modified. Note that the unprocessed measurement values are retained using the data items `_pd_meas_*`. Since the original measurements are still available, modifications like these do not result in the loss of the original data. In fact, by placing processed values in multiple blocks, a single CIF may contain measurements that have been processed in more than one way.

It is good practice to use the `_pd_proc_info_author_*` and `_pd_proc_info_datetime` data items. It is also a good idea to

describe how the measurements were processed using **_pd_proc_info_data_reduction**.

The **_pd_proc_*** data items in this list may be used to calibrate the 2θ or energy values of the data. These are defined in the items **_pd_proc_2theta_corrected**, **_pd_proc_2theta_range_***, **_pd_proc_d_spacing**, **_pd_proc_energy_***, **_pd_proc_recip_len_Q** and **_pd_proc_wavelength**.

When corrections, scaling or other processing, such as averaging or smoothing, are applied to the intensities, the results are stored using the **_pd_proc_intensity_*** data items. Note that if the number of data points does not change, it might be most convenient to include the processed intensities in the same loop as the observed values. This is not always possible, so these items can be placed in a separate loop if there is no longer a one-to-one correspondence between the 2θ or energy positions for the **_pd_proc_intensity_*** values and the **_pd_meas_counts_*** or **_pd_meas_intensity_*** values.

For energy-dispersive measurements, the incident spectrum must be determined for normalization. This can be recorded using **_pd_proc_intensity_incident**. For other types of normalization, **_pd_proc_intensity_norm** should be used.

For full-pattern fitting, there is a series of **_pd_proc_ls_*** data items for recording settings and results. For example, agreement factors can be recorded using the **_pd_proc_ls_prof_*_factor** data items. Some data items may be included in the loop(s) containing the measured or the processed data: **_pd_proc_ls_weight** specifies the weight assigned to each point and **_pd_proc_intensity_bkg_calc** specifies the fitted background. Note that background values are usually generated by extrapolation from fixed values set during the refinement or are determined from a function that is fitted to the observations, and occasionally both are used together. The function that has been fitted can be described using **_pd_proc_ls_background_function**, while fixed points are listed using **_pd_proc_intensity_bkg_fix**. If sections of the pattern are not fitted, this is indicated using **_pd_proc_info_excluded_regions**.

3.3.5.2. Simulated data

The data items in these categories are as follows:

(*a*) Part of PD_DATA

_pd_calc_intensity_net
_pd_calc_intensity_total
_pd_calc_point_id

(*b*) PD_CALC

_pd_calc_method

It is common to calculate powder-diffraction intensities from a crystallographic model. This is necessary for Rietveld refinements, where the model is fitted to the experimentally observed intensities. It is also used to simulate the diffraction pattern of a material for which the structure is known, perhaps for comparison with a measured diffraction pattern.

A crystallographic model can be described in CIF using data items from the core CIF dictionary, as described in Chapter 3.2. To record the results of the simulation, the data items **_pd_calc_intensity_net** or **_pd_calc_intensity_total** are used. The difference between these two data items depends on the treatment of background. If the pattern is simulated with a fitted background added to it, **_pd_calc_intensity_total** is used; otherwise **_pd_calc_intensity_net** is used. The values will typically be placed in a loop with the processed (**_pd_proc_***) data items or the observed (**_pd_meas_***) data items. If neither observed nor processed data are present (*e.g.* for a simulation), or if, for some reason, the simulation has been performed with a different 2θ range or step size, the appropriate **_pd_proc_*** data items are used to define the 2θ values *etc.* used for the simulation.

3.3.5.3. Diffraction peak table

The data items in these categories are as follows:

(*a*) PD_PEAK

- **_pd_peak_id**
  _pd_peak_2theta_centroid
  _pd_peak_2theta_maximum
  _pd_peak_d_spacing
  _pd_peak_intensity
  _pd_peak_pk_height
  _pd_peak_wavelength_id
  → ***_diffrn_radiation_wavelength_id***
  _pd_peak_width_2theta
  _pd_peak_width_d_spacing

(*b*) PD_PEAK_METHOD

_pd_peak_special_details

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics are defined in the core CIF dictionary.

When diffraction intensities are first measured, particularly when attempting to identify unknown phases in a material, the first step in the analysis is often to compile a list of peak positions. These peak positions are commonly used to search the Powder Diffraction File, which contains lists of peak heights and positions for approximately 100 000 materials (International Centre for Diffraction Data, 2004).

Information on diffraction peaks is recorded in the PD_PEAK section of the pdCIF. Peak positions are recorded using **_pd_peak_2theta_maximum** or **_pd_peak_2theta_centroid**, for positions determined from the intensity maxima or from the peak centroids, respectively. It is also possible to record peak positions using **_pd_peak_d_spacing**. Peak intensities are recorded using **_pd_peak_intensity** and **_pd_peak_pk_height**, for the integrated peak area or the intensity value at the peak maximum, respectively. Peak widths are recorded using **_pd_peak_width_2theta** and **_pd_peak_width_d_spacing**.

A separate loop is used to list reflections, as will be discussed in Section 3.3.5.4. To link reflections to peaks (one peak may consist of many reflections), each peak is assigned a unique code using **_pd_peak_id**, which is then referenced in the reflection table using **_pd_refln_peak_id**.

When intensities are measured using radiation with more than one wavelength, for example when both Cu $K\alpha_1$ and $K\alpha_2$ radiation are used or when a monochromator passes both λ and $\lambda/2$ radiation, peaks may be assigned a wavelength symbol using **_pd_peak_wavelength_id**, where the wavelength symbol is defined in a separate **_diffrn_radiation_wavelength_id** loop. However, for many experiments, the assignment of wavelengths to peaks will be impractical owing to reflection overlap. It is usually better practice to specify wavelength labels in the reflection table using **_pd_refln_wavelength_id**.

3.3.5.4. Reflection assignments and intensities

In addition to the REFLN data items defined in the core CIF dictionary, the following items are defined:

REFLN

_pd_refln_peak_id
→ **_pd_peak_id**
_pd_refln_phase_id
→ **_pd_phase_id**

† **_pd_refln_wavelength_id (*_refln_wavelength_id*)**
→ ***_diffrn_radiation_wavelength_id***

The arrow (→) is a reference to a parent data item. The dagger (†) indicates a deprecated item, which should not be used in the creation of new CIFs. Items in italics are defined in the core CIF dictionary.

In a single-crystal experiment, a reflection table contains the initial experimental observations for structural analysis. In contrast, the reflection table for a powder-diffraction experiment is a derived result that depends on the model used to apportion intensity between overlapping reflections. Another difference is that in a single-crystal experiment, the reflection list will refer to only one phase (one hopes), while it is common to have reflections from more than one phase in a powder-diffraction reflection list.

A list of reflections in a powder-diffraction pattern is commonly generated by Rietveld analysis, where Hugo Rietveld's algorithm (Rietveld, 1967, 1969) is used to estimate the intensity of each reflection. Alternatively, when the structure of one or more phases is not known, it is possible to use full-pattern intensity-extraction methods such as the algorithms developed by Pawley (1981) or Le Bail *et al.* (1988). In fact, intensity information obtained by full-pattern intensity extraction is often used for *ab initio* structure determination.

Most of the information in the reflection table will be defined using data items from the core CIF dictionary (see Section 3.2.2.2 and Chapter 4.1). For example, **_refln_index_h**, **_refln_index_k** and **_refln_index_l** will be used for the indices. The structure factors and reflection intensities are specified using **_refln_intensity_calc**, **_refln_intensity_meas**, **_refln_F_squared_calc** and **_refln_F_squared_meas**; reflection positions are defined using **_refln_d_spacing**. To link a reflection with a powder-diffraction peak, the pdCIF data item **_pd_refln_peak_id** is used. The value for **_pd_refln_peak_id** serves as a pointer to an entry in the peak table which has been labelled, using the data name **_pd_peak_id**, with the same symbol. Likewise, to link a reflection to a phase, the pdCIF data item **_pd_refln_phase_id** points to a phase defined using **_pd_phase_id** in the phase table. Since a single reflection may be observed with more than one wavelength, for example, with $\lambda/2$ or $K\alpha_2$ wavelengths, the pdCIF dictionary defines a wavelength link, **_pd_refln_wavelength_id**, that defines a wavelength label. However, since version 2.1, the core CIF dictionary defines **_refln_wavelength_id** and this should be used in preference to **_pd_refln_wavelength_id**. The data items **_refln_wavelength_id** and **_pd_refln_wavelength_id** both point to a wavelength label defined using **_diffrn_radiation_wavelength_id**.

The International Centre for Diffraction Data abstracts peak positions and heights for inclusion in the Powder Diffraction File. This information would be found in the **_pd_peak** section of a pdCIF. However, in many studies, particularly in Rietveld refinements, peak tables are never generated. In principle, it should be possible to calculate peak positions and peak heights (or better still, peak areas) from the information in a reflection table. An algorithm for this would be very useful.

3.3.6. Atomicity, chemistry and structure

The structural model of a compound determined by powder-diffraction methods can be described by the data items in the core CIF dictionary. However, for a powder-diffraction study of a mixture of phases, the PD_PHASE category is used to list the phases present. This is the only category in the pdCIF dictionary that extends the description of the structural model beyond that covered by items in the core CIF dictionary.

3.3.6.1. Table of phases

The data items in this category are as follows:

PD_PHASE
• **_pd_phase_id**
_pd_phase_block_id
_pd_phase_mass_%
_pd_phase_name

The bullet (•) indicates a category key.

When a sample contains more than one phase, the PD_PHASE data items are used to create a table describing the phases present. For example, the name and abundance of each phase can be specified using **_pd_phase_name** and **_pd_phase_mass_%**, respectively.

Two types of pointers can also be defined:

(i) Since the crystallographic description of each phase must be incorporated in a separate data block, **_pd_phase_block_id** contains the unique block ID (see Section 3.3.7) pointing to the block containing the data for the phase.

(ii) An arbitrary label is assigned to every phase using **_pd_phase_id** so that reflections can be assigned to a phase using **_pd_refln_phase_id**. This is discussed further in Section 3.3.5.4.

3.3.7. File metadata

The many data items in the core dictionary that decribe file auditing and history cover most of the metadata requirements of a pdCIF, but two new data items in the pdCIF category PD_BLOCK are introduced to provide a specific mechanism for identifying and relating individual data blocks.

Data items in this category are as follows:

PD_BLOCK
_pd_block_diffractogram_id
_pd_block_id

The data item **_pd_block_id** is used to define a unique name for each data block. This name is used so that one data block may reference another data block. Since CIF blocks may be separated into different files, or many CIFs from different sources may be grouped into a single file, the block ID provides a robust mechanism for maintaining references between blocks, independent of how CIF blocks have been arranged between files. The intent is that a site that archives pdCIFs will construct an index to **_pd_block_id** names that can be used to resolve block ID references.

The definition for **_pd_block_id** gives a procedure for creating a **_pd_block_id** name that is extremely unlikely to be duplicated. Other mechanisms for creating unique names can also be used: for example, using a web page name (URL) could be appropriate if care is taken never to reuse the URL.

The need for the block ID/block pointer mechanism is demonstrated by the following example. Consider a case where a neutron powder diffraction data set and an X-ray powder diffraction data set have been used together to determine a single structural model for a single crystalline phase. CIF does not allow the two data sets to be placed in a single block, since this would require two independent loops of observations where each loop uses some of the same data names. One can create a CIF with two blocks and include the structural model in the block that contains either of the two data sets. However, if this is done, a logical link is needed between the two blocks to make it clear that the structural model was derived from both data sets. It is better practice to place the

Example 3.3.7.1. *A CIF with multiple data blocks, demonstrating a suitable construction when multiple data sets and multiple phases occur together.*

```
#= First CIF block ================================
data_NISI_overall

_pd_block_id  2003-02-04T18:02|NISI|B_H_Toby|Overall

# publication and sample preparation information
# appears here (_publ_*, _journal_*, _pd_char_*
# & _pd_prep_* items are omitted for brevity)

# Overall powder R-factors
_pd_proc_ls_prof_wR_factor            0.0370
# (other _refine_ls_* items omitted for brevity)

# pointers to the phase blocks
loop_    _pd_phase_block_id
      2003-02-04T18:02|NISI_phase1|B_H_Toby||
      2003-02-04T18:02|NISI_phase2|B_H_Toby||
# pointers to the diffraction patterns
loop_    _pd_block_diffractogram_id
      2003-02-04T18:02|NISI_H_01|B_H_Toby|GPD
      2003-02-04T18:02|NISI_H_02|B_H_Toby|GPD
```

structural model in a third data block, as this emphasizes the fact that the model is derived from both data sets. Again, logical links to the data sets are needed.

In both these cases, the data item `_pd_block_diffractogram_id` would be included in the data block containing the structural model and will point to `_pd_block_id` values assigned in the data blocks containing the diffraction data to establish the connection between the data sets and the structural model. The presence of more than one value for `_pd_block_diffractogram_id`, through use of a loop, indicates that multiple data sets were used and thus these structural results are from a combined refinement. Sometimes, powder and single-crystal diffraction data are used together (most commonly to team X-ray single-crystal diffraction data with neutron powder diffraction data). In this case, `_pd_block_diffractogram_id` will point to two `_pd_block_id` values, where one is assigned to the single-crystal data set.

In contrast to the example above, in which block pointers are used to link a single structural model to multiple data sets, another application for these pointers is for describing materials that contain more than one phase. In this case, `_pd_phase_block_id` is placed in the data block containing the data set to link it to the blocks defining the phases.

In summary, three types of links between data blocks are defined.

(i) `_pd_block_diffractogram_id` connects a phase to one or more data-set blocks;

(ii) `_pd_phase_block_id` connects a data set to one or more phase blocks;

(iii) `_pd_calib_std_external_block_id` connects a block to measurements used to provide calibration constants used in the block.

It is good practice to use both `_pd_block_diffractogram_id` and `_pd_phase_block_id` in a pdCIF with multiple blocks.

3.3.7.1. Use of block pointers

More complex link structures will be needed when multiple data sets and multiple phases occur together. Example 3.3.7.1 outlines a pdCIF reporting the results of a TOF powder-diffraction study of a physical mixture of nickel and silicon powders in which two separate diffraction banks, measured at two different Bragg angles, were used. In this case, five CIF blocks are used. The first CIF

Example 3.3.7.1. (*cont.*)

```
#= Second CIF block ===============================
# Information for phase 1
data_NISI_phase_1

_pd_block_id 2003-02-04T18:02|NISI_phase1|B_H_Toby||

# Data sets for phase 1
loop_    _pd_block_diffractogram_id
  2003-02-04T18:02|NISI_H_01|B_H_Toby|GPD
  2003-02-04T18:02|NISI_H_02|B_H_Toby|GPD

_pd_phase_name                          Nickel
_cell_length_a                          3.523433(29)
_cell_length_b                          3.523433
_cell_length_c                          3.523433
_cell_angle_alpha                       90.0
_cell_angle_beta                        90.0
_cell_angle_gamma                       90.0
_cell_volume                            43.74194
_symmetry_cell_setting                  cubic
_symmetry_space_group_name_H-M          "F m 3 m"
loop_ _symmetry_equiv_pos_site_id
      _symmetry_equiv_pos_as_xyz
       1 +x,+y,+z           2 -x,-y,-z
# (other symmetry operations omitted for brevity)

loop_
     _atom_site_type_symbol
     _atom_site_fract_x
     _atom_site_fract_y
     _atom_site_fract_z
     _atom_site_occupancy
     _atom_site_thermal_displace_type
     _atom_site_U_iso_or_equiv
     _atom_site_symmetry_multiplicity
NI   0.0  0.0  0.0   1.0   Uiso 0.00435(10) 4

loop_  _atom_type_symbol  _atom_type_number_in_cell
      NI  4.0
# (_chemical_* & _geom_* items omitted for brevity)

#= Third CIF block ================================
# Information for phase 2
data_NISI_phase_2

_pd_block_id 2003-02-04T18:02|NISI_phase2|B_H_Toby||

# Data sets for phase 2
loop_    _pd_block_diffractogram_id
  2003-02-04T18:02|NISI_H_01|B_H_Toby|GPD
  2003-02-04T18:02|NISI_H_02|B_H_Toby|GPD

_pd_phase_name                          Silicon
_cell_length_a                          5.42957(9)
_cell_length_b                          5.42957
_cell_length_c                          5.42957
_cell_angle_alpha                       90.0
_cell_angle_beta                        90.0
_cell_angle_gamma                       90.0
_cell_volume                            160.06508
_symmetry_cell_setting                  cubic
_symmetry_space_group_name_H-M          "F d 3 m"
loop_ _symmetry_equiv_pos_site_id
      _symmetry_equiv_pos_as_xyz
       1 +x,+y,+z           2 -x,-y,-z
# (other symmetry operations omitted for brevity)

loop_
     _atom_site_type_symbol
     _atom_site_fract_x
     _atom_site_fract_y
     _atom_site_fract_z
     _atom_site_occupancy
     _atom_site_thermal_displace_type
     _atom_site_U_iso_or_equiv
     _atom_site_symmetry_multiplicity
SI  0.125 0.125 0.125  1.0  Uiso  0.00540(21)  8

loop_  _atom_type_symbol  _atom_type_number_in_cell
     SI  8.0
 # (_chemical_* & _geom_* items omitted for brevity)
```

Example 3.3.7.1. (*cont.*)

```
#= Fourth CIF block ================================
# Powder diffraction data for data set 1
data_NISI_p_01

_pd_block_id 2003-02-04T18:02|NISI_H_01|B_H_Toby|GPD

# (numerous _exptl_, _pd_*, _diffrn_ items describing
# the data set are omitted for brevity)

# phase table
loop_ _pd_phase_id
      _pd_phase_block_id
      _pd_phase_mass_%
      _pd_proc_ls_peak_cutoff
  1  2003-02-04T18:02|NISI_phase1|B_H_Toby||
     51(49)    0.00500
  2  2003-02-04T18:02|NISI_phase2|B_H_Toby||
     49(49)    0.00500
#  (_pd_proc_ls_profile_function omitted from loop)

loop_ _atom_type_symbol
      _atom_type_scat_length_neutron
      _atom_type_scat_source
  NI   1.0300   International_Tables_Vol_C
  SI   0.4149   International_Tables_Vol_C

_diffrn_radiation_probe                neutron
_pd_proc_ls_prof_wR_factor             0.0384
_pd_proc_ls_prof_wR_expected           0.0294
_refine_ls_R_Fsqd_factor               0.07288

_pd_proc_info_datetime          2003-02-04T18:02:09
_pd_calc_method                 "Rietveld Refinement"
_pd_meas_2theta_fixed           148.29

#---- raw data loop -----
loop_ _pd_meas_time_of_flight
      _pd_meas_intensity_total
      _pd_meas_point_id
      1000.0    1818(34)     626
# (4494 TOF & intensity values omitted for brevity)
_pd_meas_number_of_points              4495

#---- calculated data loop -----
loop_
     _pd_proc_d_spacing
     _pd_proc_intensity_total
     _pd_proc_ls_weight
     _pd_proc_intensity_bkg_calc
     _pd_calc_intensity_total
     _pd_proc_point_id
   0.50035  0.424(7)  19401.  0.3726  0.4155    1
# (1647 processed/calculated points omitted for
# brevity)
_pd_proc_number_of_points              1648

# reflection table
# Note: contains reflections for both phases
loop_
     _refln_index_h
     _refln_index_k
     _refln_index_l
     _pd_refln_phase_id
     _refln_observed_status
     _refln_F_squared_meas
     _refln_F_squared_calc
     _refln_phase_calc
     _refln_d_spacing
   4   0   0  2 o   9.773   9.812  180.00   1.35739
   3   3   1  2 o   4.799   4.801   0.00    1.24563
   2   2   0  1 o  15.254  15.195   0.00    1.24572
# (54 reflections omitted for brevity)
   4   4   4  1 o   7.498   8.733   0.00    0.50856
   9   5   3  2 o   2.350   2.396   0.00    0.50631
   8   6   4  2 o   0.000   0.000  180.00   0.50412
_reflns_number_observed                60

# (_reflns_limit_* and _reflns_d_* items omitted for
# brevity)
```

Example 3.3.7.1. (*cont.*)

```
#= Fifth CIF block =================================
# Powder diffraction data for data set 2
data_NISI_p_02

_pd_block_id 2003-02-04T18:02|NISI_H_02|B_H_Toby|GPD

# (numerous _exptl_, _pd_*, _diffrn_ items describing
# the data set are omitted for brevity)

# phase table
loop_ _pd_phase_id
      _pd_phase_block_id
      _pd_phase_mass_%
      _pd_proc_ls_peak_cutoff
  1  2003-02-04T18:02|NISI_phase1|B_H_Toby||
     51.38          0.00500
  2  2003-02-04T18:02|NISI_phase2|B_H_Toby||
     48.62(28)      0.00500
# (_pd_proc_ls_profile_function omitted from loop)

loop_ _atom_type_symbol
      _atom_type_scat_length_neutron
      _atom_type_scat_source
  NI   1.0300  International_Tables_Vol_C
  SI   0.4149  International_Tables_Vol_C

_diffrn_radiation_probe                neutron
_pd_proc_ls_prof_wR_factor             0.0363
_pd_proc_ls_prof_wR_expected           0.0222
_refine_ls_R_Fsqd_factor               0.07645

_pd_proc_info_datetime          2003-02-04T18:02:09
_pd_calc_method                 "Rietveld Refinement"
_pd_meas_2theta_fixed           88.05

#---- raw data loop -----
loop_ _pd_meas_time_of_flight
      _pd_meas_intensity_total
      _pd_meas_point_id
      750.4    2780(42)     470
# (4650 TOF & intensity values omitted for brevity)
_pd_meas_number_of_points              4651

#---- calculated data loop -----
loop_
     _pd_proc_d_spacing
     _pd_proc_intensity_total
     _pd_proc_ls_weight
     _pd_proc_intensity_bkg_calc
     _pd_calc_intensity_total
     _pd_proc_point_id
  0.45802  0.778(9)  12931.  0.4211  0.7851    1
# (1932 processed/calculated points omitted for
# brevity)
_pd_proc_number_of_points              1933

# reflection table
loop_
     _refln_index_h
     _refln_index_k
     _refln_index_l
     _pd_refln_phase_id
     _refln_observed_status
     _refln_F_squared_meas
     _refln_F_squared_calc
     _refln_phase_calc
     _refln_d_spacing
   2   0   0  1 o  16.505  16.060   0.00    1.76172
   3   1   1  2 o   4.854   5.087  180.00   1.63708
   2   2   2  2 o   0.000   0.000   0.00    1.56738
# (76 reflections omitted for brevity)
  11   3   3  2 o   1.948   2.014   0.00    0.46053
  10   6   2  2 o   0.000   0.000   0.00    0.45888
   7   3   1  1 o   7.261   7.499   0.00    0.45871
   5   5   3  1 o   7.261   7.499   0.00    0.45871
_reflns_number_observed                83

#  (_reflns_limit_* and _reflns_d_* items omitted for
# brevity)
```

block reports the overall and publication details. The next two CIF blocks report crystallographic information for each phase and the last two blocks report the observed, processed and calculated diffraction intensities and reflection tables.

A second purpose for `_pd_block_id` is to provide a mechanism for tracking successive modifications to a CIF. Consider the case where a data set is obtained at a user facility and the resulting measurements are distributed as a CIF. In this file, a value is supplied for `_pd_block_id` based on the time when the measurements were made. At a later time, when these observations are analysed, a new CIF is created, containing both the original measurements and the results from the analysis. Rather than replace the original value for `_pd_block_id`, the data item can be placed in a loop and another value, defining a second block ID, can be added. This will

indicate the connection to the initial CIF, since the original block ID is retained.

A potential future use for block pointers may be to reference non-CIF data files that contain large two- and three-dimensional data structures. This is expected to become increasingly important as neutron and synchrotron instruments are constructed that cover increasing ranges of solid angle. As mentioned in Section 3.3.2, CIF is not well suited to these complex, large and possibly irregular measurement arrays. The NeXus format has been developed by a consortium of synchrotron and neutron laboratories to address these concerns and is currently being used for a variety of scattering applications (NeXus, 1999). The NeXus format is based on the platform-independent HDF binary standard (HDF, 1998). The use of block pointers to resolve references to non-CIF documents will require additional definitions.

3.3.8. pdCIF for storing unprocessed measurements

While many researchers prepare a CIF only when a project is complete, there are good reasons for preparing a pdCIF when the diffraction data are measured, as this is the best time to document how the measurement was performed. Much of the instrumental information will remain unchanged for all pdCIFs from a given diffraction instrument, so it is a good idea to prepare a file that describes each of the common settings for an instrument. This file will probably contain some of the following data items and their associated values:

(i) The `_pd_instr_*` items, such as the instrument type in `_pd_instr_geometry`, the size of the instrument and the collimation in `_pd_instr_dist_*` and `_pd_instr_divg_*`, and monochromatization in `_pd_instr_monochr_*` (see Section 3.3.4.3)

(ii) Depending on how the calibration is performed, it may be appropriate to include `_pd_calib_*` items.

(iii) Information about the radiation source should be specified using the `_diffrn_radiation_*` and `_diffrn_source_*` data items.

(iv) Detector information should be specified using `_diffrn_detector_*` items, for example, the detector type in `_diffrn_detector_type` and perhaps calibration values such as the deadtime (in `_diffrn_detector_dtime`).

A second section of the pdCIF will contain information specific to the experiment, such as the diffraction conditions (*i.e.* pressure and temperature) recorded using the `_diffrn_ambient_*` data items. Sample and specimen information will appear in the `_pd_prep_*`, `_pd_spec_*` and `_pd_char_*` data items.

A third section of the pdCIF contains the observations. The data items used to specify the unprocessed observations will vary with the type of instrument used, as described in Sections 3.3.8.1 to 3.3.8.10 below.

3.3.8.1. Single pulse-counting detectors

In the most common measurement method, where a single pulse-counting detector is scanned over a range of 2θ, the `_pd_meas_*` entries (see Section 3.3.4.4) will be of the form shown in Example 3.3.8.1. If the data were scanned using a variable step size, the observations might be given as shown in Example 3.3.8.2. Note that when `_pd_meas_counts_*` is used, the values given must be counts, so that the standard uncertainty will be the square root of the intensity values. This means that the intensity values must not be scaled, for example if the values were counts per second; otherwise the statistical uncertainty estimates will be incorrect.

Example 3.3.8.1. *Measurements from a single pulse-counting detector with constant-step scan.*

```
_pd_meas_2theta_range_min  5.0
_pd_meas_2theta_range_max 65.0
_pd_meas_2theta_range_inc  0.02
_pd_meas_number_of_points  3001
_pd_meas_scan_method       step
_pd_meas_step_count_time   10
loop_
  _pd_meas_counts_total
  10 16 23 18 30 45 58 123 80 67 32 21 12 ...
```

Example 3.3.8.2. *Measurements from a single pulse-counting detector with variable-step scan.*

```
_pd_meas_number_of_points  3001
_pd_meas_scan_method       step
_pd_meas_step_count_time   10
loop_
  _pd_meas_2theta_scan
  _pd_meas_counts_total
  5.00 10 5.02 16 5.04 23 5.06 18 5.07 30 5.08 45
  ... ...
```

3.3.8.2. Detectors that do not count pulses

When the method used to detect intensities does not count individual quanta as they hit the detector, for example, the digitization of intensities recorded on film or on an imaging plate, or even with data recorded using a detector having a built-in deadtime correction, the standard-uncertainty values are not the square root of the intensities. [Note that when the actual deadtime correction is known, it is best to incorporate this scaling into the monitor value (see `_pd_meas_counts_monitor` in Section 3.3.4.4) or else save the uncorrected measurements and create a second set of corrected intensity values as `_pd_proc_intensity_net` (see Section 3.3.5.1).] The `_pd_meas` entries for an experiment using non-pulse-counting detection will look like the examples given in Section 3.3.8.1, except that the data loop will be in the form

```
loop_
  _pd_meas_intensity_total
  10 16 23 18 30 45 58 123 80 67 32 21 12 ...
```

or

```
loop_
  _pd_meas_2theta_scan
  _pd_meas_intensity_total
  5.00 10 5.02 16 5.04 23 5.06 18 5.07 30 5.08 45
  ... ...
```

If standard uncertainties for the intensity values are known, they can be given using the conventional notation

```
loop_
  _pd_meas_2theta_scan
  _pd_meas_intensity_total
  5.00 10(10) 5.02 16(11) 5.04 23(13) 5.06 18(12)
  5.07 30(18) ...
```

Note that when `_pd_meas_intensity_*` is used, it is best to specify `_pd_meas_units_of_intensity` as well.

3.3.8.3. Multiple detectors

At present, CIF does not offer the ability to construct true multi-dimensional data structures. However, many instruments with multiple detectors produce reasonably tractable numbers of data points. For such instruments, it is possible to include an additional data item, `_pd_meas_detector_id`, in the loop with the data to indicate the detector that made the observation.

Example 3.3.8.3. *Identifying intensities from multiple detectors.*

```
loop_
  _pd_meas_2theta_scan
  _pd_meas_intensity_total
  _pd_meas_detector_id
  5.0  10 A   25.0  16 B  45.0 23 C  65.0 18 D
  5.02 16 A   25.02 30 B  ...  ... ...
```

Example 3.3.8.4. *Measurements from an energy-dispersive X-ray diffraction experiment.*

```
_pd_meas_scan_method       disp
_pd_meas_2theta_fixed      6.5
_pd_calib_2theta_offset    0.1071
loop_
  _pd_meas_detector_id
  _pd_meas_counts_total
  _pd_proc_energy_detection
  _pd_proc_recip_len_Q
      0  180  6114.0 .714
      1  166  6141.2 .717
      2   11  6168.4 .721
      3   11  6195.5 .724
```

In Example 3.3.8.3, four detectors placed 20° apart are referenced with arbitrarily chosen labels A, B, C and D. Note that the detector characteristics will typically be specified in a separate calibration loop containing terms such as `_pd_calib_detector_id`, `_pd_calib_detector_response` and `_pd_calib_2theta_offset`. The labels given for `_pd_calib_detector_id` should match those in `_pd_meas_detector_id`.

3.3.8.4. Energy-dispersive X-ray detection

For energy-dispersive X-ray diffraction, an X-ray detector is placed at a fixed value of 2θ and a diffractogram is measured on a multichannel analyser. The channel number is then calibrated to yield photon energies. From the energy and 2θ angle, a d-spacing or Q value ($Q = 4\pi \sin\theta/\lambda$) is calculated for each diffraction point. Note that energy, d spacing or Q are not the experimental independent variable. Rather, they result from processing, since calibration information is required. The calibration equation should be described in `_pd_calibration_conversion_eqn`.

In Example 3.3.8.4, the nominal 2θ setting is 6.5°, but the actual position (determined by prior calibration) is 6.6071°, so the difference is indicated using a `_pd_calib_2theta_offset` value (see Section 3.3.4.3).

3.3.8.5. Neutron time-of-flight detection

Neutron time-of-flight (TOF) detection in theory should be no different from energy-dispersive X-ray detection, but TOF instruments record complex three-dimensional data structures, where diffraction intensities are recorded as a function of time for as many as several hundred detectors. For some instruments, both the position along the detector and the time of flight are recorded, so there may be effectively thousands of detectors. To add even further complexity, the data may be binned in different time steps for detectors at different 2θ values. CIF is likely to be cumbersome for the storage of unprocessed measurements from TOF instruments, owing to the one-dimensional nature of CIF, but it could be useful to translate files from one binary format to another using CIF as a common intermediate. To do this, a single loop is used for all data points, where each detector (or detector section, in the case of a position-sensitive detector) is assigned a detector ID. In a second loop, the detector ID values are defined. In addition to 2θ, `_pd_meas_angle_omega` and `_pd_meas_angle_chi` are defined where needed (Example 3.3.8.5).

Example 3.3.8.5. *Measurements from a neutron time-of-flight diffraction experiment.*

```
_pd_meas_scan_method       tof
loop_
  _pd_meas_detector_id
  _pd_meas_time_of_flight
  _pd_proc_d_spacing
  _pd_meas_counts_total
  _pd_proc_intensity_incident
    88  1101.6  0.251658559  11843  4003
    88  1103.2  0.25202477   11934  4001
    88  1104.8  0.252391011  11906  3999
    88  1106.4  0.252757192  11773  3997
    88  1108.0  0.253123432  11707  3995
    ...
    150 1500.0  0.257258087   6559  3185
    ...

loop_
  _pd_calib_detector_id
  _pd_meas_2theta_fixed
     88  88.05
     150 148.29
```

TOF data are usually reduced to a small number of 'banks' consisting of intensity as a function of d space or Q, where multiple detectors are summed. Data in this form can be recorded using a loop containing `_pd_proc_d_spacing` and `_pd_proc_intensity_net`. A data block is needed for each bank.

3.3.8.6. Digitized film and image plates

To record intensities from digitized X-ray film or from image plates properly requires the storing of two-dimensional data structures, which in some cases can be accommodated through imgCIF (see Chapters 2.3 and 3.7). However, it is possible to record a one-dimensional scan using `_pd_meas_position` and `_pd_meas_intensity_total` (not `_pd_meas_counts_total`!). `_pd_proc_2theta_corrected` values can then be assigned using calibration information, and they can then be included in the same loop, as in Section 3.3.8.4.

3.3.8.7. Direct background measurements

For some diffraction experiments, particularly for the determination of radial distribution functions, measurements are made for background scattering from the diffraction instrument and from the sample container. When this is done, the values can be included in a single loop using `_pd_meas_counts_background`, `_pd_meas_counts_container` and `_pd_meas_counts_total`.

3.3.8.8. Noting sample orientation

For texture measurements, intensity measurements can be made as a function of different sample setting angles. These setting angles can be specified using `_pd_meas_angle_chi`, `_pd_meas_angle_omega` and `_pd_meas_angle_phi`. The change in these values may be specified by including these data items in the loop with the diffraction intensities. In some cases, it may be more convenient to separate measurements with different setting angles into different blocks. In this case, the values for the setting angle(s) that are invariant will be set outside of a loop.

It is common in powder diffraction to reduce preferred orientation and improve crystallite averaging by rocking or rotating the sample. This is indicated by specifying the axis used for rocking, usually φ for capillary specimens or χ for flat-plate specimens, as `_pd_meas_rocking_axis`. The data item

`_pd_meas_rocking_angle` is used to record the angular range through which the sample is rocked, where 360 indicates that one complete revolution occurs during each counting period. Numbers greater than 360 are possible.

3.3.8.9. Use of an incident-intensity monitor

For radiation sources for which the intensity may vary, such as synchrotron-radiation sources, the intensity of the incident radiation is measured using an incident-intensity monitor. This value may be specified for every data point using `_pd_meas_counts_monitor` (or `_pd_meas_intensity_monitor`) by including this data item in the loop with the diffraction intensities. For some instruments, counting times are set so that the same number of monitor counts are measured for each data point. If this is the case, `_pd_meas_counts_monitor` will be the same for every data point and need not be included in the loop.

3.3.8.10. Recording detector livetime

The detector deadtime is often more a function of the counting electronics than of the intrinsic properties of the detector. In these circumstances, the counting circuit may provide a gating signal that indicates when the electronics are processing an event *versus* when the circuit is idle and waiting for an event to process. From this gating signal, a detector livetime signal can be generated. Livetime is a better way to correct intensities than applying a deadtime correction, because if appreciable numbers of events are processed but are not counted (for example, counts due to fluorescence), the actual deadtime can be quite high, even though the recorded number of counts can be quite low. To use the livetime signal, the count time can be multiplied by the livetime or the livetime can be treated as a monitor (see Section 3.3.8.9). If an incident-intensity monitor and a livetime are both available, the `_pd_meas_intensity_monitor` value can contain the incident intensity times the livetime.

3.3.9. Use of pdCIF for Rietveld refinement results

One of the major aims of the development of the pdCIF definitions was to be able to communicate the results of Rietveld refinements, and this is expected to be the most common use for pdCIF. To aid the development of software that prepares pdCIF output from Rietveld refinements, this section describes the blocks and loops to be found in a pdCIF, noting variations due to the type of Rietveld refinement. Programmers may also wish to look at the *GSAS2CIF* program, which creates CIFs for a wide range of types of diffraction data, and for multiple data sets and phases (Toby *et al.*, 2003).

It is valuable for the CIF to contain the structural model(s), the observed powder-diffraction intensities and the calculated powder-diffraction intensities so that the fit of the model to the observed diffraction pattern can be viewed graphically. It is the present author's firm belief that it is impossible to judge the quality of a Rietveld refinement by *R* factors or any other numerical metric, since these values describe not just how well the structural model fits the measurements, but also how well the background and peak shape are fitted as well. Very poor models can have good *R* factors and χ^2 values if there is a significant amount of non-Bragg scattering that has been well fitted. On the other hand, with high-resolution observations measured to excellent precision, even trivial imperfections in the peak shapes can result in poor agreement factors. There is no substitute for the visual examination of a plot of the observed and calculated patterns, optimally at more than one magnification level. The program *pdCIFplot* (Toby, 2003) plots the observed and calculated powder-diffraction intensities in a pdCIF and allows the fit to be examined in more detail than can be provided by a figure showing the whole profile at once.

3.3.9.1. A single phase

When a single set of diffraction measurements is used to model a single phase, a pdCIF will usually contain only one block. There will be several important loops present.

One loop will contain atomic parameters, such as coordinates. The unit cell must also be specified.

A second loop will contain the reflection table.

A third loop will contain the observed (or processed) diffraction measurements and the simulated pattern. Other items that should be included in this loop are the least-squares weights (usually σ^{-2}, where σ is the standard uncertainty) so that it is possible to determine the quality of the fit in individual regions. Weight values of zero can also be used to indicate that data points have been excluded from the refinement. Since background fitting is quite important in Rietveld analysis, it is also valuable to include the background values. Thus, this loop should specify:

(i) the ordinate of the Rietveld plot, using one or more of: `_pd_meas_2theta_scan`, `_pd_meas_time_of_flight`, `_pd_proc_2theta_corrected`, `_pd_proc_d_spacing` or `_pd_proc_recip_len_Q`; alternatively the ordinate can be specified using either `_pd_meas_2theta_range_*` or `_pd_proc_2theta_range_*`, where `_*` is `_min`, `_max` and `_inc` outside the loop.

It is recommended that all CIFs describing the results of a Rietveld refinement include either `_pd_proc_d_spacing` or `_pd_proc_recip_len_Q`.

(ii) The observed (or processed) intensity values, using the items `_pd_meas_counts_total`, `_pd_meas_intensity_total`, `_pd_proc_intensity_total` or `_pd_proc_intensity_net`.

(iii) The background, using the item `_pd_proc_intensity_bkg_calc`.

(iv) The least-squares weights, using the item `_pd_proc_ls_weight`. If these weights are not specified, then it must be presumed that all points have been used in the refinement and that the weights are the reciprocal of the intensity values (if `_pd_meas_counts_total` was used) or the reciprocal of the intensity standard uncertainties, if specified.

(v) The calculated pattern should appear using either `_pd_calc_intensity_net` or `_pd_calc_intensity_total`.

It is good practice always to include at least one data item from each entry in the list above.

Apart from the information contained in these loops, information from almost all sections of the pdCIF dictionary can be valuable. Such items include data items that define how the diffraction measurements were made, how the sample was prepared and characterized, how the refinement was performed, and least-squares parameters and *R* factors. A template and an example pdCIF showing the combined use of pdCIF and core data items form part of the *Acta Crystallographica* instructions for authors at http://journals.iucr.org/services/cif/powder.html.

3.3.9.2. Multiple phases

When more than one phase is present, multiple CIF blocks are needed. The resulting CIF will contain much the same information as would be found in a single-phase pdCIF, as described in Section 3.3.9.1. However, there will be a separate block for each phase containing information specific to that phase, such as the unit cell and the loop containing the atomic parameters.

The CIF will usually (see Section 3.3.7) contain one additional block with the observed and calculated pattern and a reflection table, as well as the other data items that define how the

diffraction measurements were made, how the refinement was performed *etc.* While reflection tables for each phase can be placed in each phase block, it is better to include a single reflection table in the block that contains the diffraction data. This block will also contain a phase table that uses the block pointer `_pd_block_diffractogram_id` to link to the phase blocks. The phase blocks can also be linked to the data block using the block pointer `_pd_phase_block_id`. For most Rietveld refinements, each phase is allowed to have different profile parameters, so `_pd_proc_ls_profile_function` should also be included in the phase-table loop.

3.3.9.3. One phase, multiple sets of measurements

It is fairly common to use more than one diffraction data set to determine a model for a single phase. Some examples include: combined refinement using both neutron and X-ray powder diffraction; use of multiple X-ray wavelengths to make use of anomalous dispersion; and the use of single-crystal X-ray and powder neutron diffraction data in a single refinement. For these cases, there will be a CIF block for each data set. Each of these blocks will contain a reflection table and a loop with the observed and calculated diffraction intensities, as described in Section 3.3.9.1.

As explained in Section 3.3.7, the resulting structural parameters could be placed in a block with one of the sets of diffraction data. However, it is better practice to create one additional block for these parameters, as it then becomes clear that the result is from a combined refinement. This is indicated by linking the phase block and the data-set blocks using a loop of `_pd_block_diffractogram_id` values in the phase block. The data-set blocks can also have a link to the phase information using the block pointer `_pd_phase_block_id`.

3.3.9.4. Multiple sets of measurements and phases

Multiple data sets may be used for mixtures as well as single phases. This is becoming increasingly common as more complex materials are studied using powder diffraction. The treatment of this case follows logically from that of Sections 3.3.9.2 and 3.3.9.3. If there are M diffraction data sets and P phases, there will be P blocks containing the crystallographic parameters for each phase. There will be M blocks with the observed and calculated diffraction intensities, as well as reflection tables. Depending on the Rietveld software, there may be $M \times P$ sets of some parameters, for example phase fractions and profile descriptions. These parameters may be placed in the phase table loop within the data-set block(s).

Ideally, the same specimen, or at least the same sample, will be used for all measurements. Sometimes, however, different samples are used for combined refinements to extend the number of observations, despite the possibility that the samples might have slightly different structures or compositions. If there are S samples, there will be an additional S blocks that record the sample and specimen preparation and characterization information. Thus, in this case there will be a total of $M + P + S$ blocks.

As before, the phase blocks will use the block pointers `_pd_block_diffractogram_id` to link to the data-set blocks. Likewise, the data-set blocks will have phase tables with `_pd_phase_block_id` values that link to the phase blocks. The sample blocks can use both `_pd_block_diffractogram_id` values and `_pd_phase_block_id` values to link to the the diffraction data and the analysis results. This is shown in the CIF in Example 3.3.7.1. The program *GSAS2CIF* (Toby *et al.*, 2003) can create CIFs for multiple sets of measurements and phases.

Example 3.3.10.1. *Phase identification using a reflection table and a phase table.*

```
loop_
  _pd_peak_2theta_centroid
  _pd_peak_id
     3 A1
     4 B1
     6 A2
     8 B2
     9 A3
    12 A4
    15 A5
    16 B3

loop_
  _refln_index_h
  _refln_index_k
  _refln_index_l
  _pd_refln_peak_id
  _pd_refln_phase_id
  ? ? ? A1 Phase1
  ? ? ? A2 Phase1
  ? ? ? A3 Phase1
  ? ? ? A4 Phase1
  ? ? ? A5 Phase1

loop_
  _pd_phase_id
  _pd_phase_name
  Phase1  'component 1'
```

3.3.10. Other pdCIF applications

As mentioned above, there are other applications for pdCIF than the storage of unprocessed measurements and the reporting of the results of a Rietveld refinement. This section describes the use of data items in other common pdCIF applications.

3.3.10.1. Simulated intensities

It is common to simulate a diffraction pattern from a known or hypothetical structural model. The structural model is recorded in CIF using core data items, such as `_atom_site_label`, `_atom_site_fract_x`, `_atom_site_fract_y`, `_atom_site_fract_z`, `_atom_site_U_iso_or_equiv` and `_atom_site_occupancy`, as well as the unit cell in `_cell_length_*` and `_cell_angle_*`. Calculated reflection intensities can be recorded using `_refln_index_*` and `_refln_F_squared_calc`, as described in Section 3.3.5.4. The simulated pattern can be recorded using `_pd_calc_*` data items, as described in Section 3.3.5.2.

The simulated diffraction pattern will be determined not only by the structural parameters, but also by the type of experiment that is being simulated. For example, it is good practice to define data items to specify the type of radiation in `_diffrn_radiation_probe`, the wavelength in `_diffrn_radiation_wavelength` and the profile in `_pd_proc_ls_profile_function`.

3.3.10.2. Phase identification and indexing

For phase identification, a CIF will include unprocessed measurements, as described in Section 3.3.8. Sample characterization information, for example chemical analysis information, can often aid phase determination. Characterization information is described in Section 3.3.4.1. Similarly, sample preparation information can also be quite valuable (see Section 3.3.4.1). Since preferred orientation or other artifacts of the measurement can make phase

Table A3.3.1.1. *Categories in the powder CIF dictionary*
Numbers in parentheses refer to the section of this chapter where each category is described in detail.

PD_BLOCK group (§3.3.7)	PD_MEAS group (§3.3.4.4)
PD_BLOCK (§3.3.7)	PD_DATA (*part of category*) (§3.3.4.4(*a*))
PD_CALC group (§3.3.5.2)	PD_MEAS_INFO (§3.3.4.4(*b*))
PD_CALC (§3.3.5.2(*b*))	PD_MEAS_METHOD (§3.3.4.4(*c*))
PD_DATA (*part of category*) (§3.3.5.2(*a*))	PD_PEAK group (§3.3.5.3)
PD_CALIB group (§3.3.4.3)	PD_PEAK (§3.3.5.3(*a*))
PD_CALIB (§3.3.4.3(*a*))	PD_PEAK_METHOD (§3.3.5.3(*b*))
PD_CALIBRATION (§3.3.4.3(*b*))	PD_PHASE group (§3.3.6.1)
PD_CHAR group (§3.3.4.1)	PD_PHASE (§3.3.6.1)
PD_CHAR (§3.3.4.1(*a*))	PD_PREP group (§3.3.4.1)
PD_DATA group	PD_PREP (§3.3.4.1(*b*))
_pd_calc_* items (§3.3.5.2(*a*))	PD_PROC group (§3.3.5.1)
_pd_instr_var_illum_len (§3.3.4.3(*d*))	PD_DATA (*part of category*) (§3.3.5.1(*a*))
_pd_meas_* items (§3.3.4.4(*a*))	PD_PROC_INFO (§3.3.5.1(*b*))
_pd_proc_* items (§3.3.5.1(*a*))	PD_PROC_LS (§3.3.5.1(*c*))
PD_INSTR group (§3.3.4.3)	PD_SPEC group (§3.3.4.2)
PD_DATA (*part of category*) (§3.3.4.3(*d*))	PD_SPEC (§3.3.4.2)
PD_INSTR (§3.3.4.3(*c*))	REFLN group (§3.3.5.4)
	REFLN (§3.3.5.4)

identification more difficult, it is a good idea to include specimen preparation and mounting information, as described in Section 3.3.4.2.

Peaks will be located and documented in the peak table, as discussed in Section 3.3.5.3. In this case, it can be very helpful to specify **_pd_peak_wavelength_id** for peaks that are clearly a $K\alpha_2$ component. Similarly, recording a peak width can also be helpful for some autoindexing programs.

To identify peaks by phase in the case where at least one phase in a material is known but other peaks remain unidentified requires the use of a reflection table and a phase table, as shown in Example 3.3.10.1. This example shows a diffraction pattern with eight peaks, of which five have been identified as arising from a phase where the unit cell has not yet been determined. The peaks labelled B1, B2 and B3 are referenced in the peak table, but are not defined in the reflection loop. This implies that they arise from an unknown phase or phases. The remaining peaks A1 to A5 are referenced in the peak table. Note that the reflection table must include the **_refln_index_*** data items, even though no reflection indices are assigned. This is because CIF rules require that the **_refln_index_*** data items be present in this loop, as noted in the pdCIF dictionary definitions for **_pd_refln_peak_id** and **_pd_refln_phase_id**. The place holder **?** is used to indicate that the reflection indices are not yet known.

Appendix 3.3.1
Category structure of the powder CIF dictionary

Table A3.3.1.1 provides an overview of the structure of the powder CIF dictionary by informal category group.

Many people contributed to the creation of this dictionary, most notably Syd Hall, I. David Brown and Brian McMahon. Ian Langford helped considerably with the evolution of the dictionary. This project also benefited from the efforts of many members of the PDF-3 Database Format Subcommittee of the International Centre for Diffraction Data; I am particularly indebted to Walter Schriener for useful concepts and discussions. My initial involvement with powder diffraction file formats was prompted by the encouragement of Richard Harlow; I may someday forgive him.

References

Finger, L. W., Cox, D. E. & Jephcoat, A. P. (1994). *A correction for powder diffraction peak asymmetry due to axial divergence. J. Appl. Cryst.* **27**, 892–900.

HDF (1998). *NCSA HDF home page*. http://hdf.ncsa.uiuc.edu/.

International Centre for Diffraction Data (2004). Powder Diffraction File. International Centre for Diffraction Data, 12 Campus Boulevard, Newtown Square, Pennsylvania, USA. http://www.icdd.com.

Le Bail, A., Duroy, H. & Fourquet, J. L. (1988). *Ab initio structure determination of $LiSbWO_6$ by X-ray powder diffraction*. *Mater. Res. Bull.* **23**, 447–452.

NeXus (1999). *NeXus data format*. http://www.nexus.anl.gov.

Pawley, G. S. (1981). *Unit-cell refinement from powder diffraction scans. J. Appl. Cryst.* **14**, 357–361.

Rietveld, H. M. (1967). *Line profiles of neutron powder-diffraction peaks for structure refinement. Acta Cryst.* **22**, 151–152.

Rietveld, H. M. (1969). *A profile refinement method for nuclear and magnetic structures. J. Appl. Cryst.* **2**, 65–71.

Toby, B. H. (2003). *CIF applications. XII. Inspecting Rietveld fits from pdCIF: pdCIFplot. J. Appl. Cryst.* **36**, 1285–1287.

Toby, B. H., Von Dreele, R. B. & Larson, A. C. (2003). *CIF applications. XIV. Reporting of Rietveld results using pdCIF: GSAS2CIF. J. Appl. Cryst.* **36**, 1290–1294.

3.4. Classification and use of modulated and composite structures data

BY G. MADARIAGA

3.4.1. Introduction

Aperiodic structures do not have lattice periodicity, but do show long-range order. Their diffraction patterns exhibit sharp Bragg peaks that have to be indexed using more than three Miller indices. All aperiodic structures can be classified into one of three groups: incommensurately modulated structures, incommensurate composite structures and quasicrystals. It is the purpose of the modulated and composite structures CIF dictionary (msCIF dictionary), presented in Chapter 4.3, to provide machine-readable descriptions of the first two of these groups.

Modulated structures represent the simplest case and are described by periodic distortions of an underlying (reference) crystal structure. The distortions involve one or several atomic parameters – positions, occupation probability or thermal parameters (the term *displacement parameters* is ambiguous in this context) – and their periodicity may be commensurate or incommensurate with the lattice of the reference structure. The first case corresponds to a three-dimensionally periodic crystalline phase, whereas the second case defines an incommensurate structure. In both cases, the diffracted intensities can be divided into two groups: the prominent main reflections, which are located in a three-dimensional reciprocal lattice (that of the reference structure), and additional (generally) weaker satellite reflections situated at points determined by the wave vectors of the modulations. Strictly speaking, the number of parameters needed to describe the atomic modulations of an incommensurate structure is infinite. In practice, however, the number of structural parameters is often drastically reduced owing to the strong hierarchy (demonstrated by the discreteness of the diffraction diagrams) that exists among them.

Composite structures can be thought of as being built of two or more periodic subsystems whose lattices are mutually incommensurate. Therefore, the set of main reflections no longer defines a unique three-dimensional reciprocal lattice. Moreover, the interaction between the different subsystems provokes modulations and, as a consequence, the presence of satellite reflections in the diffraction diagram.

Quasicrystals not only lack three-dimensional lattice periodicity but also show noncrystallographic symmetry.

The methodology for solving aperiodic crystal structures has been well developed since the introduction of the concept of superspace (de Wolff, 1974; de Wolff *et al.*, 1981). Superspace allows the recovery of the periodicity and a simple description of the symmetry of quasicrystalline structures in a higher-dimensional space. The real aperiodic structure is recovered from the superspace through appropriate three-dimensional sections. The dimension of the superspace $(3 + d)$ is equal to the number of reciprocal vectors needed to index the whole diffraction pattern of the quasicrystalline structure. More information on the superspace approach can be found, for example, in van Smaalen (1995) and Janssen *et al.* (2004).

Affiliation: GOTZON MADARIAGA, Departamento de Física de la Materia Condensada, Facultad de Ciencia y Tecnología, Universidad del País Vasco, Apartado 644, 48080 Bilbao, Spain.

The success of this concept relies above all on the systematic description of the symmetry of the aperiodic materials using superspace groups. Superspace groups for the simplest (but most common) case of one-dimensional modulated structures are tabulated in Janssen *et al.* (2004).

Within the superspace approach, all the aperiodic atom positions are embedded in dense sets (atomic domains) in the $(3 + d)$-dimensional unit cell of the associated periodic structure in superspace. They are parallel on average to the 'internal' (or 'perpendicular') space, which is a d-dimensional subspace chosen to be orthogonal to the real (physical or parallel) space. The three-dimensional structure is then a section of the $(3+d)$-dimensional structure parallel to the real space. The atomic domains are distorted along the internal space by the modulation functions. In many incommensurate structures and composites, the atomic domains are continuous periodic functions along the internal subspace and are parameterized by Fourier series. Some compounds, however, need to be described using discrete atomic domains whose parameterization is more complicated and will be discussed in Section 3.4.2. Note that commensurate structures can also be included within the superspace approach. The difference between incommensurate and commensurate modulations is that for commensurate modulations only a finite number of values of the atomic modulation functions are relevant.

The number of modulated and composite structures solved with software that uses the superspace formalism has grown rapidly and in many cases the determination of the structures of such systems is now almost routine. A standard for the description of incommensurate modulated structures has been established by the Commission on Aperiodic Crystals of the International Union of Crystallography (Chapuis *et al.*, 1997) using a checklist that is easily extensible to composite crystals. However, there was until recently no standard way to represent these structures electronically. Structural databases tend to contain only a brief reference to the modulated character of the structure and it has not been possible to transmit, archive and retrieve information about modulated and composite structures as efficiently as for normal crystal structures. Extending the core CIF dictionary to form a modulated and composite structures CIF (msCIF) dictionary seemed the appropriate way to deal with these problems, and has the additional benefits derived from the use of a well tested standard for which several tools have been developed.

In the case of quasicrystals, however, although the theoretical foundations seem to be well established, the determination of accurate models requires a combination of different strategies and techniques. The parameterization of the atomic domains with physical meaning is far from being an automated procedure (Cervellino *et al.*, 2002) and some of the existing models are now disputed. Major problems arise from sample quality, intrinsic disorder and rather low data-to-parameter ratios (Haibach *et al.*, 2000). Quasicrystals are not covered by the msCIF dictionary detailed in Chapter 4.3.

3.4.2. Dictionary design considerations

The CIF dictionary for modulated and composite structures (msCIF dictionary) is an extension of the core CIF dictionary

Table 3.4.2.1. *Category groups defined in the msCIF dictionary*

The groups are listed in the order in which they are described in this chapter.

Section	Category group	Subject covered
(*a*) Experimental measurements		
3.4.3.1.1	CELL	The unit cell, especially cell subsystems for composite structures
3.4.3.1.2	DIFFRN	Multi-dimensional diffraction pattern
3.4.3.1.3	EXPTL	Details of the experiment specific to modulated and composite structures
(*b*) Analysis		
3.4.3.2.1	REFINE	Refinement procedures
3.4.3.2.2	REFLN	Reflection measurements indexed in higher-dimensional space
(*c*) Atomicity, chemistry and structure		
3.4.3.3.1 to 3.4.3.3.3	ATOM	Atom sites in a modulated structure
3.4.3.3.4	GEOM	Geometry of a structure in superspace
3.4.3.3.5	SYMMETRY	Symmetry information
(*d*) File metadata		
3.4.3.4	AUDIT	The structure of the CIF

(Chapter 4.1). This means that the dictionary defines items that are basically related to single-crystal data. The close relationship between the msCIF and core dictionaries has led to synergies between and benefits for both dictionaries.

The design of the msCIF dictionary had two objectives: (i) it should be as functional as possible, *i.e.* as little information in an msCIF as possible should be given as unstructured text; (ii) it should be possible to include even the oldest modulated and composite structures in an msCIF, even if the way they were reported did not follow the guidelines used now.

There were two major difficulties in implementing the msCIF dictionary. Firstly, the number of additional wave vectors used to index a diffraction diagram is theoretically not limited. Secondly, a CIF containing information about a modulated or composite structure should, in general, be composed of several (related) data blocks. As CIF definitions do not at present include vectors or matrices as distinct types of data values, an arbitrary upper limit of 11 was assigned to the dimension of superspace to limit the number of new data names. Linking between data blocks is handled by using recommended values for items in the AUDIT and AUDIT_LINK categories, like those used in the powder CIF (pdCIF) dictionary (Chapters 3.3 and 4.2).

An additional problem arises when special (ideal) modulation functions are considered. Although periodic modulations are normally parameterized by Fourier series, in certain cases it is convenient to use discontinuous functions which lead to a severe reduction in the number of structural parameters. The shape of these functions is not restricted and new materials could require new functions. Given that it is not possible at this moment to define logical or mathematical relations between data values within a CIF [although an initiative for including algorithms in the definitions of CIF dictionaries has been proposed by Spadaccini *et al.* (2000)], general functions cannot be defined and therefore the type of special functions included in the msCIF dictionary are those implemented in the most widely used program, *JANA2000* (Petříček & Dušek, 2000). They only apply to one-dimensional modulations and are sawtooth displacive functions and occupational crenel functions. Both functions define discontinuous occupational atomic domains and are normally combined with (smoother) atomic modulation functions (involving atom positions and/or thermal parameters) that are expressed by Fourier series. Because of the discreteness of the atomic domains, members of the set of harmonic functions used to expand these series are no longer mutually orthogonal as they are only defined within each atomic domain and not in the (internal space) interval $[0, 1]$. As a consequence, severe correlation effects among the coefficients of the Fourier series are expected. A solution for this problem lies in the selection and orthogonalization of a set of basic functions (Petříček *et al.*, 1995). The atomic modulation functions are then expressed as linear combinations of an orthogonal basis whose elements are specific combinations of harmonic functions. Discontinuous atomic domains are being increasingly used in composite materials, in some cases revealing that considering such materials as composites or modulated structures is a matter of convenience (Elcoro *et al.*, 2003; Pérez-Mato *et al.*, 2003).

CIFs that conform to the msCIF dictionary are highly itemized for the human reader, but have a strong relational structure even though the dictionary itself is written in DDL1.

The major drawbacks of the dictionary are:

(i) Items describing the superspace symmetry should be reconsidered and perhaps included within the symmetry CIF (symCIF) dictionary (Chapters 3.8 and 4.7).

(ii) There is still some information in an msCIF that cannot be interpreted by a computer [*e.g.* rigid rotations and translations around (along) noncrystallographic axes cannot be parsed, since the description of the axes is textual].

(iii) A full description of the modulation in terms of orthogonalized functions (used when the atomic domains are discrete) is not supported yet.

3.4.3. Arrangement of the dictionary

The msCIF dictionary detailed in Chapter 4.3 includes 19 new categories. Another 18 already exist in the core CIF dictionary, but include new items (16) or revised definitions (2). The category structure of the msCIF dictionary is summarized in Table 3.4.2.1 and is listed in full in Appendix 3.4.1. The appendix also lists for each category the section of this chapter in which the category is described.

Many of the modifications to categories that already exist in the core CIF dictionary result from the need to use more than three integer indices to label the diffracted intensities (in the cases of CELL, DIFFRN_REFLN, DIFFRN_REFLNS, DIFFRN_STANDARD_REFLN and EXPTL_CRYSTAL_FACE) or the need to use superspace symmetry (in the cases of GEOM_ANGLE, GEOM_BOND, GEOM_CONTACT, GEOM_TORSION, SPACE_GROUP and SPACE_GROUP_SYMOP). Apart from the categories that describe the atomic modulation functions, there are two that are specific to composite structures (CELL_SUBSYSTEM and CELL_SUBSYSTEMS).

The rest of this section summarizes the contents of the dictionary, organized by categories within the functional groups outlined in Table 3.1.10.1. As in the other chapters in this part of the volume, the classification is under the headings *Experimental measurements* (Section 3.4.3.1), *Analysis* (Section 3.4.3.2), *Atomicity, chemistry and structure* (Section 3.4.3.3) and *File metadata* (Section 3.4.3.4). The msCIF dictionary adds no new data items concerned with the publication or reporting of structures to those already present in the core CIF dictionary.

The data items within each category are listed in the detailed commentary below. Where relevant, data items that represent a unique identifier for a looped list ('category keys') are listed first and are marked by a bullet (•). The remaining data items in each category are listed alphabetically.

3.4.3.1. Experimental measurements

The categories relevant to a structure determination experiment are in the CELL, DIFFRN and EXPTL groups. Most of the items extend the existing core CIF categories, but the categories describing cell subsystems are new.

3.4.3.1.1. *Cell and modulation wave vectors*

The categories describing the unit cell (or cells for composite structures), the wave vectors of the modulations and, for composites, the cell subsystems are as follows:

CELL group
- CELL ¶
- CELL_SUBSYSTEM
- CELL_SUBSYSTEMS
- CELL_WAVE_VECTOR
- CELL_WAVE_VECTORS

Categories marked with ¶ are already defined in the core CIF dictionary.

The data items in these categories are as follows:

(*a*) CELL
- **_cell_modulation_dimension**
- **_cell_reciprocal_basis_description**

(*b*) CELL_SUBSYSTEM
- **_cell_subsystem_code**
- **_cell_subsystem_description**
- **_cell_subsystem_matrix_W_1_1**
- ⋮
- (and **_cell_subsystem_matrix_W_*m*_*n*** for all combinations $1 \leq m,n \leq 11$)
- ⋮
- **_cell_subsystem_matrix_W_11_11**

(*c*) CELL_SUBSYSTEMS
- **_cell_subsystems_number**

(*d*) CELL_WAVE_VECTOR
- **_cell_wave_vector_seq_id**
- **_cell_wave_vector_x**
- **_cell_wave_vector_y**
- **_cell_wave_vector_z**

(*e*) CELL_WAVE_VECTORS
- **_cell_wave_vectors_meas_details**
- **_cell_wave_vectors_pressure_max**
- **_cell_wave_vectors_pressure_min**
- **_cell_wave_vectors_temp_max**
- **_cell_wave_vectors_temp_min**
- **_cell_wave_vectors_variation**

As explained in Section 3.4.2, the msCIF dictionary arbitrarily allows an upper limit of 11 for the dimension of superspace for which data names are defined. **_cell_modulation_dimension** specifies the number of additional reciprocal vectors needed to index the whole diffraction pattern and has values d ranging from 1 to 8 to express the dimensionality $(3 + d)$ of the superspace. **_cell_reciprocal_basis_description** is a text field allowing a free description of the higher-dimensional basis chosen.

For a composite structure, different cell subsystems may be specified. Each such subsystem is identified and characterized by the data items in the CELL_SUBSYSTEM category (see Section 3.4.4.1). **_cell_subsystems_number** gives the number of such subsystems as an independent check of the completeness of the description.

Data items in the CELL_WAVE_VECTOR category specify the wave vectors of the modulation (see Section 3.4.4.1). In accordance with the limits on dimensionality of the current version of the msCIF dictionary, no more than eight additional modulation wave vectors may be specified. The number used must agree with the value of **_cell_modulation_dimension**.

The data items in the CELL_WAVE_VECTORS category describe the experimental conditions during the determination of the independent modulation wave vectors.

3.4.3.1.2. *Data collection*

The categories describing data collection are as follows:

DIFFRN group
- DIFFRN_REFLN ¶
- DIFFRN_REFLNS ¶
- DIFFRN_STANDARD_REFLN ¶

Categories marked with ¶ are already defined in the core CIF dictionary.

New data items in these categories are as follows:

(*a*) DIFFRN_REFLN
- **_diffrn_refln_index_m_1**
- **_diffrn_refln_index_m_2**
- **_diffrn_refln_index_m_3**
- **_diffrn_refln_index_m_4**
- **_diffrn_refln_index_m_5**
- **_diffrn_refln_index_m_6**
- **_diffrn_refln_index_m_7**
- **_diffrn_refln_index_m_8**

(*b*) DIFFRN_REFLNS
- **_diffrn_reflns_limit_index_m_1_max**
- **_diffrn_reflns_limit_index_m_1_min**
- **_diffrn_reflns_limit_index_m_2_max**
- **_diffrn_reflns_limit_index_m_2_min**
- **_diffrn_reflns_limit_index_m_3_max**
- **_diffrn_reflns_limit_index_m_3_min**
- **_diffrn_reflns_limit_index_m_4_max**
- **_diffrn_reflns_limit_index_m_4_min**
- **_diffrn_reflns_limit_index_m_5_max**
- **_diffrn_reflns_limit_index_m_5_min**
- **_diffrn_reflns_limit_index_m_6_max**
- **_diffrn_reflns_limit_index_m_6_min**
- **_diffrn_reflns_limit_index_m_7_max**
- **_diffrn_reflns_limit_index_m_7_min**
- **_diffrn_reflns_limit_index_m_8_max**
- **_diffrn_reflns_limit_index_m_8_min**
- **_diffrn_reflns_satellite_order_max**

(*c*) DIFFRN_STANDARD_REFLN
- **_diffrn_standard_refln_index_m_1**
- **_diffrn_standard_refln_index_m_2**
- **_diffrn_standard_refln_index_m_3**
- **_diffrn_standard_refln_index_m_4**
- **_diffrn_standard_refln_index_m_5**
- **_diffrn_standard_refln_index_m_6**
- **_diffrn_standard_refln_index_m_7**
- **_diffrn_standard_refln_index_m_8**

The data items in these categories are straightforward extensions of the core CIF dictionary definitions to the indexing of diffraction intensities by higher-dimensional components. The **_diffrn_refln_index_m_*** items are the additional Miller indices m_i indexing the modulation wave vectors $\mathbf{q}_i$ when the diffraction wave vector is written as $\mathbf{H} = h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^* + \sum_i m_i\mathbf{q}_i$.

The upper limit of 8 was chosen arbitrarily to limit the number of data items defined in the msCIF dictionary.

_diffrn_reflns_limit_index_m_* items provide independent checks on the range of values recorded for each of the additional Miller indices m_i. **_diffrn_standard_refln_index_m_*** items allow the higher-dimensional Miller indices of standard reflections to be recorded.

3.4.3.1.3. *Experimental measurements on the crystal*

The categories describing measurements on the crystal or crystals used in the experiment are as follows:

EXPTL group
EXPTL_CRYSTAL ¶
EXPTL_CRYSTAL_FACE ¶

Categories marked with ¶ are already defined in the core CIF dictionary.

New data items in these categories are as follows:

(*a*) EXPTL_CRYSTAL
_exptl_crystal_type_of_structure

(*b*) EXPTL_CRYSTAL_FACE
_exptl_crystal_face_index_m_1
_exptl_crystal_face_index_m_2
_exptl_crystal_face_index_m_3
_exptl_crystal_face_index_m_4
_exptl_crystal_face_index_m_5
_exptl_crystal_face_index_m_6
_exptl_crystal_face_index_m_7
_exptl_crystal_face_index_m_8

_exptl_crystal_type_of_structure specifies the structure type as `cryst` (crystalline), `mod` (modulated) or `comp` (composite). These are the only three types of structure handled at present by the msCIF dictionary.

The extensions to the EXPTL_CRYSTAL_FACE category permit the indexing of crystal faces using the higher-dimensional Miller indices introduced for aperiodic structures.

3.4.3.2. Analysis

The categories related to refinement that have been extended in this dictionary are as follows:

Refinement techniques and results (§3.4.3.2.1)
REFINE ¶
The reflections used in the refinement (§3.4.3.2.2)
REFLN ¶
REFLNS ¶

Categories marked with ¶ are already defined in the core CIF dictionary.

3.4.3.2.1. *Refinement techniques and results*

New data items in this category are as follows:

REFINE
_refine_ls_mod_func_description
_refine_ls_mod_hydrogen_treatment
_refine_ls_mod_overall_phason_coeff
_refine_ls_mod_overall_phason_formula

During the early stages of the development of the msCIF dictionary, several sets of data items were defined to accommodate the need to specify residual *R* factors for the different sets of main reflections and satellite reflections. It was then recognized that the binning of reflection classes had more general application, and these new data items were transferred to the core CIF dictionary, where, of course, they are still available for use in an msCIF.

The new items in the REFINE category in the msCIF dictionary are specific to the refinement of modulated structures. **_refine_ls_mod_func_description** allows a free-text description of the types of modulation present in the structural model and how they are handled. The treatment of hydrogen-atom modulation parameters is specified by **_refine_ls_mod_hydrogen_treatment**. Information on an overall phason correction (the use of which should in general be discouraged) may be given using the **_refine_ls_mod_overall_phason_*** items.

3.4.3.2.2. *The reflections used in the refinement*

New data items in these categories are as follows:

(*a*) REFLN
_refln_index_m_1
_refln_index_m_2
_refln_index_m_3
_refln_index_m_4
_refln_index_m_5
_refln_index_m_6
_refln_index_m_7
_refln_index_m_8

(*b*) REFLNS
_reflns_limit_index_m_1_max
_reflns_limit_index_m_1_min
_reflns_limit_index_m_2_max
_reflns_limit_index_m_2_min
_reflns_limit_index_m_3_max
_reflns_limit_index_m_3_min
_reflns_limit_index_m_4_max
_reflns_limit_index_m_4_min
_reflns_limit_index_m_5_max
_reflns_limit_index_m_5_min
_reflns_limit_index_m_6_max
_reflns_limit_index_m_6_min
_reflns_limit_index_m_7_max
_reflns_limit_index_m_7_min
_reflns_limit_index_m_8_max
_reflns_limit_index_m_8_min

As with the **_diffrn_refln_*** and **_diffrn_reflns_*** items (Section 3.4.3.1.2), these data names extend the corresponding core data items into the higher-dimensional space used in the treatment of modulated structures and composites. They apply to the list of reflections used in the refinement, as distinct from the experimentally collected set of intensities described by the **_diffrn_*** data items.

3.4.3.3. Atomicity, chemistry and structure

The categories relevant to the description of the structural model are as follows:

ATOM group
Atom sites (§3.4.3.3.1)
ATOM_SITE ¶
ATOM_SITE_PHASON
Modulation functions as Fourier series (§3.4.3.3.2)
ATOM_SITE_DISPLACE_FOURIER
ATOM_SITE_DISPLACE_FOURIER_PARAM
ATOM_SITE_FOURIER_WAVE_VECTOR
ATOM_SITE_OCC_FOURIER
ATOM_SITE_OCC_FOURIER_PARAM
ATOM_SITE_ROT_FOURIER
ATOM_SITE_ROT_FOURIER_PARAM
ATOM_SITE_U_FOURIER
ATOM_SITE_U_FOURIER_PARAM
ATOM_SITES_DISPLACE_FOURIER
ATOM_SITES_MODULATION
ATOM_SITES_ROT_FOURIER
Special modulation functions (§3.4.3.3.3)
ATOM_SITE_DISPLACE_SPECIAL_FUNC
ATOM_SITE_OCC_SPECIAL_FUNC
Molecular or packing geometry (§3.4.3.3.4)
GEOM group
GEOM_ANGLE ¶
GEOM_BOND ¶
GEOM_CONTACT ¶
GEOM_TORSION ¶
Symmetry information (§3.4.3.3.5)
SYMMETRY group
SPACE_GROUP ¶
SPACE_GROUP_SYMOP ¶

Categories marked with ¶ are already defined in the core CIF dictionary.

Most of the new categories introduced to the msCIF dictionary appear here, since their function is to describe in great detail the

modulation of the atom-site properties. They fall naturally into families describing the modulation of atomic displacement, of site occupation or of thermal parameters.

New data items are added to several categories in the core CIF dictionary that describe molecular or packing geometry. There are also new data items to describe superspace-group symmetry.

3.4.3.3.1. *Atom sites*

Data items in these categories are as follows:

(*a*) ATOM_SITE
```
  _atom_site_displace_modulation_flag
  _atom_site_occ_modulation_flag
  _atom_site_subsystem_code
        → _cell_subsystem_code
  _atom_site_U_modulation_flag
```

(*b*) ATOM_SITE_PHASON
```
• _atom_site_phason_atom_site_label
        → _atom_site_label
  _atom_site_phason_coeff
  _atom_site_phason_formula
```

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

The ATOM_SITE category is extended in the msCIF dictionary by the addition of a small number of items that may appear in the main looped list of atom-site information (see Section 3.2.4.1.1). The `*_flag` items indicate whether each individual atom site has been modelled through modulation of atomic displacement, site occupation or thermal parameters. In each case, the default value of the item is `no`, so that any or all of the flags may be omitted when that particular type of modulation has not been applied to the structural model.

`_atom_site_subsystem_code` identifies the cell subsystem to which the atom site must be assigned in the description of composite structures. Each value of `_atom_site_subsystem_code` must match one of the values of `_cell_subsystem_code` in the overall description of the subsystems defined for a composite.

The ATOM_SITE_PHASON category allow details of an atom-dependent phason correction, as implemented in *JANA2000*, to be given. The use of these phason corrections is discouraged.

3.4.3.3.2. *Modulation functions as Fourier series*

Data items in these categories are as follows:

(*a*) ATOM_SITE_DISPLACE_FOURIER
```
• _atom_site_displace_Fourier_id
  _atom_site_displace_Fourier_atom_site_label
        → _atom_site_label
  _atom_site_displace_Fourier_axis
  _atom_site_displace_Fourier_wave_vector_seq_id
        → _atom_site_Fourier_wave_vector_seq_id
```

(*b*) ATOM_SITE_DISPLACE_FOURIER_PARAM
```
• _atom_site_displace_Fourier_param_id
        → _atom_site_displace_Fourier_id
  _atom_site_displace_Fourier_param_cos
  _atom_site_displace_Fourier_param_modulus
  _atom_site_displace_Fourier_param_phase
  _atom_site_displace_Fourier_param_sin
```

(*c*) ATOM_SITE_FOURIER_WAVE_VECTOR
```
  _atom_site_Fourier_wave_vector_description
  _atom_site_Fourier_wave_vector_seq_id
  _atom_site_Fourier_wave_vector_x
  _atom_site_Fourier_wave_vector_y
  _atom_site_Fourier_wave_vector_z
```

(*d*) ATOM_SITE_OCC_FOURIER
```
• _atom_site_occ_Fourier_id
  _atom_site_occ_Fourier_atom_site_label
        → _atom_site_label
  _atom_site_occ_Fourier_wave_vector_seq_id
        → _atom_site_Fourier_wave_vector_seq_id
```

(*e*) ATOM_SITE_OCC_FOURIER_PARAM
```
• _atom_site_occ_Fourier_param_id
        → _atom_site_occ_Fourier_id
  _atom_site_occ_Fourier_param_cos
  _atom_site_occ_Fourier_param_modulus
  _atom_site_occ_Fourier_param_phase
  _atom_site_occ_Fourier_param_sin
```

(*f*) ATOM_SITE_ROT_FOURIER
```
• _atom_site_rot_Fourier_id
  _atom_site_rot_Fourier_atom_site_label
        → _atom_site_label
  _atom_site_rot_Fourier_axis
  _atom_site_rot_Fourier_wave_vector_seq_id
        → _atom_site_Fourier_wave_vector_seq_id
```

(*g*) ATOM_SITE_ROT_FOURIER_PARAM
```
• _atom_site_rot_Fourier_param_id
        → _atom_site_rot_Fourier_id
  _atom_site_rot_Fourier_param_cos
  _atom_site_rot_Fourier_param_modulus
  _atom_site_rot_Fourier_param_phase
  _atom_site_rot_Fourier_param_sin
```

(*h*) ATOM_SITE_U_FOURIER
```
• _atom_site_U_Fourier_id
  _atom_site_U_Fourier_atom_site_label
        → _atom_site_label
  _atom_site_U_Fourier_tens_elem
  _atom_site_U_Fourier_wave_vector_seq_id
        → _atom_site_Fourier_wave_vector_seq_id
```

(*i*) ATOM_SITE_U_FOURIER_PARAM
```
• _atom_site_U_Fourier_param_id
        → _atom_site_U_Fourier_id
  _atom_site_U_Fourier_param_cos
  _atom_site_U_Fourier_param_modulus
  _atom_site_U_Fourier_param_phase
  _atom_site_U_Fourier_param_sin
```

(*j*) ATOM_SITES_DISPLACE_FOURIER
```
  _atom_sites_displace_Fourier_axes_description
```

(*k*) ATOM_SITES_MODULATION
```
  _atom_sites_modulation_global_phase_t_1
  _atom_sites_modulation_global_phase_t_2
  _atom_sites_modulation_global_phase_t_3
  _atom_sites_modulation_global_phase_t_4
  _atom_sites_modulation_global_phase_t_5
  _atom_sites_modulation_global_phase_t_6
  _atom_sites_modulation_global_phase_t_7
  _atom_sites_modulation_global_phase_t_8
```

(*l*) ATOM_SITES_ROT_FOURIER
```
  _atom_sites_rot_Fourier_axes_description
```

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

It is common to represent a modulated structure using a reference periodic structure on which are superimposed atomic modulation functions expanded as Fourier series. (A full discussion of this is given in Section 3.4.4.3.) The msCIF dictionary provides separate categories for listing the modulated parameters that apply to atom positions, site occupancies and thermal parameters. The structuring of the data items within each of these categories follows a similar pattern.

For example, consider the modulation of the atomic displacements. The ATOM_SITE_DISPLACE_FOURIER category allows a listing of the axis along which the displacement occurs (`*_axis`) and the wave vectors contributing to that displacement component (`*_wave_vector_seq_id`) for each relevant atom site (labelled by `_atom_site_displace_Fourier_atom_site_label`).

_wave_vector_seq_id** is a pointer to the description of the separate modulation wave vectors and must match one of the identifiers **_atom_site_Fourier_wave_vector_seq_id** listed separately in the ATOM_SITE_FOURIER_WAVE_VECTOR category. Likewise, the ***_atom_site_label** data item must match a value of **_atom_site_label** in the main list of atom positions. This is how the modulation is linked to the atom list. The item **_atom_site_displace_Fourier_id** is the formal key for the ATOM_SITE_DISPLACE_FOURIER category. It is used to locate the matching Fourier coefficients in the ATOM_SITE_DISPLACE_FOURIER_PARAM category. The coefficients may be reported in a sine–cosine (**_atom_site_displace_Fourier_param_sin**, ***_cos**) or modulus–argument (_mod**, ***_phase**) representation.

Where a group of atoms is treated as a rigid group, the categories above describe only the translational part of the positional distortion. ATOM_SITE_ROT_FOURIER and ATOM_SITE_ROT_FOURIER_PARAM are used to describe the rotational components.

ATOM_SITE_OCC_FOURIER and ATOM_SITE_U_FOURIER, and their associated *_PARAM categories, are the analogous categories for the modulation of site occupation and thermal parameters.

All the categories above describe the properties of individual atom sites. Larger-scale descriptions of the displacive modulation or of the rotational component of a rigid group are covered by the categories ATOM_SITES_DISPLACE_FOURIER and ATOM_SITES_ROT_FOURIER, each of which at present contains one descriptive data item.

The ATOM_SITES_MODULATION category contains data items describing the initial phases of the modulation waves, which are essential for determining the space group of the commensurate superstructure. More details are given in the dictionary.

3.4.3.3.3. *Special modulation functions*

Data items in these categories are as follows:

(*a*) ATOM_SITE_DISPLACE_SPECIAL_FUNC

```
• _atom_site_displace_special_func_atom_site_label
        → _atom_site_label
  _atom_site_displace_special_func_sawtooth_ax
  _atom_site_displace_special_func_sawtooth_ay
  _atom_site_displace_special_func_sawtooth_az
  _atom_site_displace_special_func_sawtooth_c
  _atom_site_displace_special_func_sawtooth_w
```

(*b*) ATOM_SITE_OCC_SPECIAL_FUNC

```
• _atom_site_occ_special_func_atom_site_label
        → _atom_site_label
  _atom_site_occ_special_func_crenel_c
  _atom_site_occ_special_func_crenel_w
```

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

Several data items cover modulation functions that are not expressed as Fourier expansions. The examples in the current msCIF dictionary are restricted to the one-dimensional modulations (sawtooth displacive and occupational crenel functions) implemented in the program *JANA2000* (see Section 3.4.2).

3.4.3.3.4. *Molecular or packing geometry*

New data items in these categories are as follows:

(*a*) GEOM_ANGLE

```
_geom_angle_av
_geom_angle_max
_geom_angle_min
_geom_angle_site_ssg_symmetry_1
_geom_angle_site_ssg_symmetry_2
_geom_angle_site_ssg_symmetry_3
```

(*b*) GEOM_BOND

```
_geom_bond_distance_av
_geom_bond_distance_max
_geom_bond_distance_min
_geom_bond_site_ssg_symmetry_1
_geom_bond_site_ssg_symmetry_2
```

(*c*) GEOM_CONTACT

```
_geom_contact_distance_av
_geom_contact_distance_max
_geom_contact_distance_min
_geom_contact_site_ssg_symmetry_1
_geom_contact_site_ssg_symmetry_2
```

(*d*) GEOM_TORSION

```
_geom_torsion_av
_geom_torsion_max
_geom_torsion_min
_geom_torsion_site_ssg_symmetry_1
_geom_torsion_site_ssg_symmetry_2
_geom_torsion_site_ssg_symmetry_3
_geom_torsion_site_ssg_symmetry_4
```

For each of the geometry categories, there are two groups of extensions. One set covers maximum, minimum and average values of bonds, contact distances, angles and torsion angles. The other extends the symmetry-operation code used in geometry listings in the core CIF dictionary (see Section 3.2.4.3.2) to the higher-dimensional superspace form.

3.4.3.3.5. *Symmetry information*

New data items in these categories are as follows:

(*a*) SPACE_GROUP

```
_space_group_ssg_IT_number
_space_group_ssg_name
_space_group_ssg_name_IT
_space_group_ssg_name_WJJ
_space_group_ssg_WJJ_code
```

(*b*) SPACE_GROUP_SYMOP

```
_space_group_symop_ssg_id
_space_group_symop_ssg_operation_algebraic
```

At present, the msCIF dictionary extends the core CIF dictionary symmetry categories to describe superspace groups for one-dimensional modulated structures in four ways: as the superspace-group number in Janssen *et al.* (2004) (**_space_group_ssg_IT_number**), as the *International Tables* superspace-group symbol (***_ssg_name_IT**), as one of the notations from de Wolff *et al.* (1981) (***_ssg_name_WJJ**, ***_ssg_WJJ_code**), or in some other formalism (***_ssg_name**). At present, superspace-group names for higher dimensions can only be indicated using **_space_group_ssg_name**.

Symmetry operations in the superspace group are specified in the SPACE_GROUP_SYMOP category by an obvious extension to the method used in the core dictionary. These items must always be present in a CIF corresponding to a modulated or composite structure.

3.4.3.4. File metadata

The categories modified in the msCIF dictionary to formalize the construction of a multi-block description of modulated or composite structures are as follows:

AUDIT group

AUDIT ¶

AUDIT_LINK ¶

Categories marked with ¶ are already defined in the core CIF dictionary.

Data items revised in these categories are as follows:

(*a*) AUDIT
```
_audit_block_code
```
(*b*) AUDIT_LINK
```
_audit_link_block_code
```

The core dictionary definitions of these items are revised in order to formalize the relationships between multiple data blocks representing reference and modulated structures. Guidance is provided in the msCIF dictionary on how to label data blocks in a way that makes their mutual relationships clear.

3.4.4. Use of the msCIF dictionary

In this section, some of the capabilities of the dictionary will be demonstrated using simple examples. More detailed examples can be found at http://www.iucr.org/iucr-top/cif/ms and on the CD-ROM accompanying this volume.

3.4.4.1. Description of reciprocal space

Modulated and composite structures need more than three reciprocal vectors in order to index the whole set of reflections with integer numbers. Hence a diffraction vector is written as

$$\mathbf{H} = h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^* + m_1\mathbf{q}_1 + \ldots + m_d\mathbf{q}_d, \qquad (3.4.4.1)$$

where the notation has been chosen according to the core CIF dictionary. In the case of a modulated structure, $\mathbf{a}^*$, $\mathbf{b}^*$ and $\mathbf{c}^*$ are the reciprocal vectors of the reference structure (and therefore h, k and l index the main reflections). $\mathbf{q}_1, \ldots, \mathbf{q}_d$ are the modulation wave vectors. They are three-dimensional vectors with some irrational component (if the modulated structure is incommensurate) in the lattice spanned by $\mathbf{a}^*$, $\mathbf{b}^*$ and $\mathbf{c}^*$. d is the dimension of the modulation. In the case of composite structures, the diffraction pattern can be indexed using $3 + d$ (arbitrarily selected) vectors $\mathbf{a}_k^*$ ($k = 1, \ldots, 3+d$). $\mathbf{a}_1^*$ ($\equiv \mathbf{a}^*$), $\mathbf{a}_2^*$ ($\equiv \mathbf{b}^*$) and $\mathbf{a}_3^*$ ($\equiv \mathbf{c}^*$) normally span the reciprocal lattice of the main reflections of one of the substructures (notice that this is only one particular, but highly intuitive, choice). The remaining d vectors with $k = 4, \ldots, d$ are the wave vectors of the modulation [$\mathbf{q}_1, \ldots, \mathbf{q}_d$ in equation (3.4.4.1)].

In a composite structure, the $(3 + d)$-dimensional reciprocal basis of the subsystem ν is determined by a $(3 + d) \times (3 + d)$ matrix W^ν [see van Smaalen (1995) and references therein]:

$$\mathbf{a}_i^{*\nu} = \sum_{k=1}^{3+d} W_{ik}^\nu \mathbf{a}_k^*, \quad i = 1, \ldots, 3 + d, \qquad (3.4.4.2)$$

where the subscripts $i = 1$, 2 and 3 label the reciprocal vectors $\mathbf{a}^{*\nu}$, $\mathbf{b}^{*\nu}$ and $\mathbf{c}^{*\nu}$, and $i = 4, \ldots, d$ label the wave vectors of the modulation expressed as linear combinations of $\mathbf{a}^{*\nu}$, $\mathbf{b}^{*\nu}$ and $\mathbf{c}^{*\nu}$.

The simplest case corresponds to a one-dimensional ($d = 1$) modulated structure. Consider for example the incommensurate phase of K_2SeO_4. The wave vector of the modulation can be chosen to be $\mathbf{q}_1 = \alpha\mathbf{a}^*$. Relevant information about the diffraction pattern of this compound is expressed using both the core CIF and msCIF dictionaries as shown in Example 3.4.4.1.

A more complicated example is the composite structure $(LaS)_{1.14}NbS_2$. The two mutually incommensurate subsystems (along the a axis) are (van Smaalen, 1991) NbS_2 ($\nu = 1$) and LaS ($\nu = 2$). The reciprocal basis can be chosen to be $\mathbf{a}_1^* = \mathbf{a}^{*1}$, $\mathbf{a}_2^* = \mathbf{b}^{*1}$, $\mathbf{a}_3^* = \mathbf{c}^{*1}$ and $\mathbf{a}_4^* = \mathbf{a}^{*2}$. For this particular choice, the two W matrices [see equation (3.4.4.2)] are

Example 3.4.4.1. *msCIF description of the diffraction pattern of a one-dimensional modulated structure.*

```
_exptl_crystal_type_of_structure         'mod'
_cell_reciprocal_basis_description
; a*,b*,c* (reciprocal basis spanning the lattice of
  main reflections), q modulation wave vector.
;

_diffrn_symmetry_description
; The whole diffraction pattern shows orthorhombic
  symmetry. The following extinction rules were
  detected:
     0kl0   k+l=odd
     h0lm   h+m=odd
     hk0m   m=odd
     h00m   h,m=odd
  Superspace group: P:Pnam:-1ss
;

_diffrn_reflns_satellite_order_max           1

_diffrn_reflns_theta_max                  40.14
_diffrn_reflns_theta_min                   3.32
_diffrn_reflns_limit_h_max                    8
_diffrn_reflns_limit_k_max                   18
_diffrn_reflns_limit_l_max                   10
_diffrn_reflns_limit_index_m_1_max            1
_diffrn_reflns_limit_h_min                    0
_diffrn_reflns_limit_k_min                    0
_diffrn_reflns_limit_l_min                    0
_diffrn_reflns_limit_index_m_1_min           -1

# Modulation wave vector
  loop_
     _cell_wave_vector_seq_id
     _cell_wave_vector_x
         1        0.318(5)
```

$$W^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad W^2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

This information is transcribed to CIF format as shown in Example 3.4.4.2. (Note that the default values for the wave vector components and the elements of W are 0.)

3.4.4.2. Description of symmetry

The symmetry of a modulated or composite structure is described by a superspace group which leaves the $(3 + d)$-dimensional embedding of the structure invariant. Superspace is built of two orthogonal subspaces and both of them are kept invariant separately by the superspace symmetry operations. (In reciprocal space this means that, for these structures, main reflections and satellite reflections are never transformed into one another by superspace symmetry operations.) Consequently, superspace groups are not general $(3 + d)$-dimensional space groups. The standard notation for superspace groups only covers the one-dimensional superspace groups, which are listed in Janssen *et al.* (2004). As a consequence, msCIFs must include a list of all the symmetry operations in an (x, y, z) format (using as symbols $x_1 \ldots x_{3+d}$) similar to that used in the core CIF dictionary. Superspace-group names for one-dimensional structures can be expressed either according to Janssen *et al.* (2004) or according to the original notation of de Wolff *et al.* (1981). Alternative names or higher-dimensional superspace groups can also be included, but not parsed.

Example 3.4.4.2. *Representation of two mutually incommensurate subsystems.*

```
_exptl_crystal_type_of_structure        'comp'
_cell_subsystems_number                  2
_cell_modulation_dimension               1

_cell_reciprocal_basis_description
; a1*,b1*,c1* (reciprocal basis spanning the
  reciprocal basis of the first subsystem),
  a2* reciprocal axis corresponding to the
  second subsystem.
;

loop_
  _cell_subsystem_code
  _cell_subsystem_description
  _cell_subsystem_matrix_W_1_1
  _cell_subsystem_matrix_W_1_4
  _cell_subsystem_matrix_W_2_2
  _cell_subsystem_matrix_W_3_3
  _cell_subsystem_matrix_W_4_1
  _cell_subsystem_matrix_W_4_4
     NbS2  '1st subsystem'  1 0 1 1 0 1
     LaS   '2nd subsystem'  0 1 1 1 1 0

_diffrn_symmetry_description
; The whole diffraction pattern shows orthorhombic
  symmetry. The following extinction rules were
  detected:
     hklm   h+k+m=odd
     hklm   h+l=odd
     hklm   k+l+m=odd
     hk0m   h+k=odd
     hk0m   m=odd
  Extinction rules are compatible with the superspace
  groups:
     P:Fmmm:-11s(\a,0,0)
     P:Fm2m:-1-1s(\a,0,0)
;
```

Example 3.4.4.3. *Symmetry description of a superspace group.*

```
_space_group_ssg_name_WJJ   'P:P n a m:-1 s s'
_space_group_ssg_name_IT    'P n m a (\a 0 0) 0 s s'

loop_
     _space_group_symop_ssg_id
     _space_group_symop_ssg_operation_algebraic
     1   x1,x2,x3,x4
     2   1/2+x1,1/2-x2,1/2-x3,x4
     3   1/2-x1,1/2+x2,-x3,1/2-x4
     4   -x1,-x2,1/2+x3,1/2-x4
     5   -x1,-x2,-x3,-x4
     6   1/2-x1,1/2+x2,1/2+x3,-x4
     7   1/2+x1,1/2-x2,x3,1/2+x4
     8   x1,x2,1/2-x3,1/2+x4
```

In the particular case of K_2SeO_4, the superspace group is $P^{Pnma}_{\bar{1}\,s\,s}$ (de Wolff *et al.*, 1981) or $Pnma(\alpha 00)0ss$ (Janssen *et al.*, 2004). This information would appear in a CIF as shown in Example 3.4.4.3.

Example 3.4.4.4. *Atomic displacements as translations and rigid rotations.*

```
loop_
  _atom_site_description
  _atom_site_label
  _atom_site_occupancy
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
; SeO4 is a tetrahedral group formed by the atoms
  Se, O1, O2, O3 and O1p. O1p is related to O1 by the
  mirror plane perpendicular to the c axis. The
  coordinates given by _atom_site_fract_ correspond
  to the centre of mass.
;
     SeO4 1.0  0.22346    0.41868    0.250000
  .  Se   1.0  0.22216(4) 0.41861(3) 0.250000
  .  K1   1.0  0.1670(1)  0.08069(7) 0.250000
  .  K2   1.0 -0.0013(1)  0.70766(4) 0.250000
  .  O1   1.0  0.2890(3)  0.3413(2)  0.0277(3)
  .  O2   1.0  0.3130(4)  0.5621(2)  0.250000
  .  O3   1.0  0.0092(4)  0.4304(3)  0.250000

_refine_ls_mod_func_description
; Displacive modulation. Fourier series. Modulation
  of SeO4 group described in terms of rigid
  translations and rotations.
;

loop_
  _atom_site_Fourier_wave_vector_seq_id
  _atom_site_Fourier_wave_vector_x
  _atom_site_Fourier_wave_vector_description
     1   0.318(5)   'First harmonic'

loop_
  _atom_site_displace_Fourier_id
  _atom_site_displace_Fourier_atom_site_label
  _atom_site_displace_Fourier_axis
  _atom_site_displace_Fourier_wave_vector_seq_id
     K1_z_1   K1     z   1
     K2_z_1   K2     z   1
     SeO_z_1  SeO4   z   1

loop_
  _atom_site_displace_Fourier_param_id
  _atom_site_displace_Fourier_param_cos
  _atom_site_displace_Fourier_param_sin
     K1_z_1    0.0080(4)  -0.0106(5)
     K2_z_1    0.0159(4)   0.0071(6)
     SeO_z_1  -0.0089(2)  -0.0058(2)

loop_
  _atom_site_rot_Fourier_id
  _atom_site_rot_Fourier_atom_site_label
  _atom_site_rot_Fourier_axis
  _atom_site_rot_Fourier_wave_vector_seq_id
     SeO_x_1   SeO4   x   1
     SeO_y_1   SeO4   y   1

loop_
  _atom_site_rot_Fourier_param_id
  _atom_site_rot_Fourier_param_cos
  _atom_site_rot_Fourier_param_sin
     SeO_x_1  -4.2(1)   0.91(3)
     SeO_y_1   4.3(1)   0.
```

3.4.4.3. Description of the structure

A modulated structure is described by a reference periodic structure and the atomic modulation functions. Such functions are periodic and are normally expanded as Fourier series. The modulated parameters may apply to the atom positions (displacive modulation), the site occupancies (occupational modulation) and/or the temperature factors. In composite structures, each substructure is referred to the crystallographic basis defined by the W matrices [see equation (3.4.4.2)]. The simplest case corresponds to a one-dimensional displacive modulated structure. In this case, the atomic modulation functions are given by

$$u^{\mu}_{\alpha} = \sum_{n=1}^{\infty} U^{\mu}_{n\alpha} \cos\left(2\pi n\mathbf{q}\cdot\mathbf{r} + \varphi^{\mu}_{n\alpha}\right), \qquad (3.4.4.3)$$

where $(U^{\mu}_{n\alpha}, \varphi^{\mu}_{n\alpha})$ is the complex amplitude of each Fourier term; μ labels the atoms; $\alpha = x, y, z$; $\mathbf{r}$ is the average atom position; and $n\mathbf{q}$ represents the successive harmonics of the modulation. At present, displacive modulations along axes other than a, b and c can be calculated with the restriction stated in Section 3.4.2.

Example 3.4.4.5. *Relationship between data blocks in an msCIF.*

```
_audit_block_code      1997-07-24|LaSNbS2|G.M.|

loop_
  _audit_link_block_code
  _audit_link_block_description
    .   'common experimental and publication data'
    1997-07-24|LaSNbS2|G.M.|_REFRNCE
        'reference structure (global data)'
    1997-07-21|LaSNbS2|G.M.|_MOD
        'modulated structure (global data)'
    1997-07-24|LaSNbS2|G.M.|_REFRNCE_NbS2
        'reference structure (1st subsystem)'
    1997-07-21|LaSNbS2|G.M.|_MOD_NbS2
        'modulated structure (1st subsystem)'
    1997-07-24|LaSNbS2|G.M.|_REFRNCE_LaS
        'reference structure (2nd subsystem)'
    1997-07-21|LaSNbS2|G.M.|_MOD_LaS
        'modulated structure (2nd subsystem)'
```

Atomic displacements can also be expressed as rigid rotations and translations. The incommensurate phase of K_2SeO_4 is one-dimensional displacive and sinusoidal (at least over a wide range of temperature), *i.e.* $n = 1$ in equation (3.4.4.3). The tetrahedral SeO_4 groups behave as rigid bodies. Symmetry considerations restrict the possible translations to occur along **c** where the only allowed rigid rotations are around the *a* and *b* axes. The incommensurate structure is then expressed as shown in Example 3.4.4.4.

Alternatively, equation (3.4.4.3) can be expressed as a Fourier series with real amplitudes. This form is also covered by the msCIF dictionary. Note that there is a global phase, which is irrelevant in the incommensurate case but fixes the space group if the modulated structure is commensurate. Global phases are also defined in the dictionary.

3.4.4.4. Block pointers

CIFs for modulated and (in particular) composite structures often need several linked data blocks. For example, the average structure of a modulated phase refined using only the main reflections could be reported in one data block and then the results of the refinement of the complete structure using all the reflections could be added as a separate data block at a later stage. Similarly, each substructure in a composite structure can be fully described in a separate data block. It is therefore essential to keep all these blocks together, since all of them together describe the composite structure. Block pointers are described in the core CIF dictionary, but they need to be used with particular care in msCIFs. Therefore, the msCIF dictionary gives additional guidelines for naming the data blocks, as the aim (as in pdCIF) is to define unique names. Example 3.4.4.5 shows how the structure of $(LaS)_{1.14}NbS_2$ can be represented using several data blocks. In this example, the block named `1997-07-24|LaSNbS2|G.M.|` is linked to those in the loop below.

3.4.4.5. Other information

The msCIF and core CIF dictionaries also include items for *R* factors for different reflection classes (for example, for the main reflections and for the satellite reflections; see Section 3.2.3) and for describing the variation of interatomic distances due to the modulation. msCIF can be used to create a complete record of the determination of a modulated or composite structure suitable for submitting as an article to an academic journal or for archiving in a database. The obvious next step in the development of msCIF is to adapt or write software that can filter and manage msCIF data and that can generate msCIFs automatically.

Appendix 3.4.1
Category structure of the msCIF dictionary

Table A3.4.1.1 shows all categories that are represented in the msCIF dictionary.

Table A3.4.1.1. *Categories in the modulated structures CIF dictionary*

Numbers in parentheses refer to the section of this chapter in which each category is described in detail.

ATOM group (§3.4.4.3.3)	AUDIT group (§3.4.4.3.4)
ATOM_SITE (§3.4.4.3.3.1(*a*)) ¶	AUDIT (§3.4.4.3.4(*a*)) ‡
ATOM_SITE_DISPLACE_FOURIER (§3.4.4.3.3.2(*a*))	AUDIT_LINK (§3.4.4.3.4(*b*)) ‡
ATOM_SITE_DISPLACE_FOURIER_PARAM (§3.4.4.3.3.2(*b*))	CELL group (§3.4.4.3.1.1)
ATOM_SITE_DISPLACE_SPECIAL_FUNC (§3.4.4.3.3.3(*a*))	CELL (§3.4.4.3.1.1(*a*)) ¶
ATOM_SITE_FOURIER_WAVE_VECTOR (§3.4.4.3.3.2(*c*))	CELL_SUBSYSTEM (§3.4.4.3.1.1(*b*))
ATOM_SITE_OCC_FOURIER (§3.4.4.3.3.2(*d*))	CELL_SUBSYSTEMS (§3.4.4.3.1.1(*c*))
ATOM_SITE_OCC_FOURIER_PARAM (§3.4.4.3.3.2(*e*))	CELL_WAVE_VECTOR (§3.4.4.3.1.1(*d*))
ATOM_SITE_OCC_SPECIAL_FUNC (§3.4.4.3.3.3(*b*))	CELL_WAVE_VECTORS (§3.4.4.3.1.1(*e*))
ATOM_SITE_PHASON (§3.4.4.3.3.1(*b*))	DIFFRN group (§3.4.4.3.1.2)
ATOM_SITE_ROT_FOURIER (§3.4.4.3.3.2(*f*))	DIFFRN_REFLN (§3.4.4.3.1.2(*a*)) ¶
ATOM_SITE_ROT_FOURIER_PARAM (§3.4.4.3.3.2(*g*))	DIFFRN_REFLNS (§3.4.4.3.1.2(*b*)) ¶
ATOM_SITE_U_FOURIER (§3.4.4.3.3.2(*h*))	DIFFRN_STANDARD_REFLN (§3.4.4.3.1.2(*c*)) ¶
ATOM_SITE_U_FOURIER_PARAM (§3.4.4.3.3.2(*i*))	EXPTL group (§3.4.4.3.1.3)
ATOM_SITES_DISPLACE_FOURIER (§3.4.4.3.3.2(*j*))	EXPTL_CRYSTAL (§3.4.4.3.1.3(*a*)) ¶
ATOM_SITES_MODULATION (§3.4.4.3.3.2(*k*))	EXPTL_CRYSTAL_FACE (§3.4.4.3.1.3(*b*)) ¶
ATOM_SITES_ROT_FOURIER (§3.4.4.3.3.2(*l*))	GEOM group (§3.4.4.3.3.4)
	GEOM_ANGLE (§3.4.4.3.3.4(*a*)) ¶
	GEOM_BOND (§3.4.4.3.3.4(*b*)) ¶
	GEOM_CONTACT (§3.4.4.3.3.4(*c*)) ¶
	GEOM_TORSION (§3.4.4.3.3.4(*d*)) ¶
	REFINE group (§3.4.4.3.2.1)
	REFINE (§3.4.4.3.2.1) ¶
	REFLN group (§3.4.4.3.2.2)
	REFLN (§3.4.4.3.2.2(*a*)) ¶
	REFLNS (§3.4.4.3.2.2(*b*)) ¶
	SYMMETRY group (§3.4.4.3.3.5)
	SPACE_GROUP (§3.4.4.3.3.5(*a*)) ¶
	SPACE_GROUP_SYMOP (§3.4.4.3.3.5(*b*)) ¶

¶ Categories already defined in the core CIF dictionary but containing new items.
‡ A stricter definition than that given in the core CIF dictionary.

The msCIF dictionary exists thanks to the continuous help and support received from I. David Brown and Brian McMahon. The suggestions and criticisms of Sander van Smaalen, Howard Flack, Juan Manuel Pérez-Mato and, especially, Vaclav Petříček (who has implemented a CIF input/output routine in the program *JANA2000*) were important.

References

Cervellino, A., Haibach, T. & Steurer, W. (2002). *Structure solution of the basic decagonal Al–Co–Ni phase by the atomic surfaces modelling method. Acta Cryst.* B**58**, 8–33.

Chapuis, G., Farkas-Jahnke, M., Pérez-Mato, J. M., Senechal, M., Steurer, W., Janot, C., Pandey, D. & Yamamoto, A. (1997). *Checklist for the description of incommensurate modulated crystal structures. Report of the International Union of Crystallography Commission on Aperiodic Crystals. Acta Cryst.* A**53**, 95–100.

Elcoro, L., Pérez-Mato, J. M., Darriet, J. & El Abed, A. (2003). *Superspace description of trigonal and orthorhombic $A_{1+x}A'_xB_{1-x}O_3$ compounds as modulated layered structures; application to the refinement of trigonal $Sr_6Rh_5O_{15}$. Acta Cryst.* B**59**, 217–233.

Haibach, T., Cervellino, A., Estermann, M. A. & Steurer, W. (2000). *X-ray structure determination of quasicrystals – limits and potentiality. Z. Kristallogr.* **215**, 569–583.

Janssen, T., Janner, A., Looijenga-Vos, A. & de Wolff, P. M. (2004). *Incommensurate and commensurate modulated structures. International Tables for Crystallography*, Volume C, *Mathematical, chemical and physical tables*, 3rd ed., edited by E. Prince, ch. 9.8. Dordrecht: Kluwer Academic Publishers.

Pérez-Mato, J. M., Etrillard, J., Kiat, J. M., Liang, B. & Lin, C. T. (2003). *Competition between composite and modulated configurations in $Bi_2Sr_2CaCu_2O_{8+\delta}$ and its relation to oxygen stoichiometry. Phys. Rev. B*, **67**, 24504-1–24504-13.

Petříček, V. & Dušek, M. (2000). *The Crystallographic Computing System JANA*. Institute of Physics, Prague, Czech Republic.

Petříček, V., van der Lee, A. & Evain, M. (1995). *On the use of crenel functions for occupationally modulated structures. Acta Cryst.* A**51**, 529–535.

Smaalen, S. van (1991). *Superspace-group approach to the modulated structure of the inorganic misfit layer compound $(LaS)_{1.14}NbS_2$. J. Phys. Condens. Matter*, **3**, 1247–1263.

Smaalen, S. van (1995). *Incommensurate crystal structures. Crystallogr. Rev.* **4**, 79–202.

Spadaccini, N., Hall, S. R. & Castleden, I. R. (2000). *Relational expressions in STAR File dictionaries. J. Chem. Inf. Comput. Sci.* **40**, 1289–1301.

Wolff, P. M. de (1974). *The pseudo-symmetry of modulated crystal structures. Acta Cryst.* A**30**, 777–785.

Wolff, P. M. de, Janssen, T. & Janner, A. (1981). *The superspace groups for incommensurate crystal structures with a one-dimensional modulation. Acta Cryst.* A**37**, 625–636.

3.5. Classification and use of electron density data

BY P. R. MALLINSON AND I. D. BROWN

3.5.1. Introduction

Modern high-resolution X-ray diffraction methods enable the electron density distributions in many crystals to be accurately determined, allowing one to explore the interactions that hold crystals and molecules together (Coppens, 1997; Koritsanszky & Coppens, 2001). For example, Bader's (1990) theory of atoms in molecules gives a description of chemical bonding in terms of the topology of the electron density, but it requires an accurate knowledge of the electron density. [Note that the terms 'electron density' (or distribution) and 'charge density' tend to be used interchangeably in the literature. The term 'electron density' is used here since 'charge density' strictly includes both positive (nuclear) and negative (electronic) charge. Properties such as electrostatic potential, electric fields and their gradients, and molecular electric moments are derived from the charge density distribution, which includes the contribution of the nuclear charge (Spackman & Brown, 1994).] The advent of area detectors, more powerful laboratory computing facilities and developments in software have combined to produce a rapid increase in the number of accurate electron density determinations now available. While such studies are not yet routine, they are sufficiently common to justify defining a structure for a secure and systematic archive. This in turn requires a method of representing the electron density acceptable to the crystallographic community.

The CIF standard which has been adopted by the IUCr and is in common use in macromolecular and small-molecule crystallography is the obvious choice for such a file structure. At the Sagamore XIII conference held at Stare Jablonki, Poland, in 2000, the IUCr Commission on Charge, Spin and Momentum Densities decided to prepare a CIF dictionary for reporting electron densities in crystals. The result is the rhoCIF dictionary described in this chapter. It was prepared in conjunction with members of the Commission and other members of the community. [The name rhoCIF for this dictionary was chosen over the more obvious choice of edCIF to avoid any possible confusion with electron diffraction. The symbol $\rho(\mathbf{r})$ conventionally represents the electron density in real space at the point defined by the position vector $\mathbf{r}$.]

The distribution of electron density in a molecule or crystal can also be calculated theoretically, and the results are usually compared with experiment. The Hohenberg–Kohn theorem (Hohenberg & Kohn, 1964) shows that the ground-state energy is a unique functional of the electron density, thus providing a theoretical justification for relating chemical and physical properties directly to the observable electron density. Density functional theory has been applied to chemical problems for some time (see, for example, Labanowski & Andzelm, 1991; Ziegler, 1991; Kohn *et al.*, 1996; Nagy, 1998). Although the current version of the rhoCIF dictionary is intended to record the results of experimental measurements, it can also be used to represent a theoretical electron density that has been fitted with an appropriate model.

Affiliations: PAUL R. MALLINSON, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, Scotland; I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1.

3.5.2. Dictionary design considerations

The rhoCIF dictionary contains the definitions of data items used to report accurate electron densities in crystals. The structure of these definitions conforms to the CIF rules and conventions, and selected data items reflect those currently needed in the field. In any field of science, the nature of the data used to describe the experiment and the results changes with time, and accurate electron density studies are no exception. The items defined in this dictionary describe the density representation approach used most often today – the atom-centred multipole expansion model introduced by Stewart (1973) and refined by Hansen & Coppens (1978). In this model, the electron density is expressed as the sum of three components. Two are the partial contributions of the core and valence shell of the spherical free atom, usually taken from the paper by Clementi & Roetti (1974). The third component is an aspherical multipole expansion of the remaining electron density. This is an atom-centred function, flexible enough to describe most electron density distributions likely to be encountered in a crystal, and with components that have the same symmetries as the atom-centred wave functions (orbitals) frequently used by chemists when describing chemical bonding.

3.5.3. Classification of data definitions

The rhoCIF dictionary is intended to be used in combination with the core CIF dictionary, or, after adaptation to DDL2, the mmCIF dictionary. The two dictionaries can be combined to form a composite dictionary by concatenation. Alternatively, the merging procedures described in Section 3.1.9 can be used to create a virtual dictionary at run time. Because the dictionary is not self-contained, its structure can be kept very simple, comprising only two categories: one to define a local axis system around each atom and one to contain the various parameters of the Hansen & Coppens (1978) multipole function.

The two categories in the electron density CIF dictionary lie within the single ATOM category group and relate to the structural model. They are as follows:

Specification of local axes at each atom (§3.5.3.1)
ATOM_LOCAL_AXES

Atom-centred multipole expansion functions (§3.5.3.2)
ATOM_RHO_MULTIPOLE

3.5.3.1. Specification of local axes at each atom

Data items in this category are as follows:

ATOM_LOCAL_AXES
- **_atom_local_axes_atom_label**
  → **_atom_site_label**

_atom_local_axes_atom0
_atom_local_axes_atom1
_atom_local_axes_atom2
_atom_local_axes_ax1
_atom_local_axes_ax2

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

Example 3.5.3.1. *Definition of the local Cartesian axis system for the atoms in a proton sponge complex.*

```
loop_
_atom_local_axes_atom_label
_atom_local_axes_atom0
_atom_local_axes_ax1
_atom_local_axes_atom1
_atom_local_axes_atom2
_atom_local_axes_ax2
N(1)    C(1)    X   N(1)    N(2)    Y
N(2)    C(8)    X   N(2)    N(1)    Y
C(1)    N(1)    X   C(1)    C(2)    Y
C(2)    C(1)    X   C(2)    C(3)    Y
C(3)    C(2)    X   C(3)    C(4)    Y
C(4)    C(10)   X   C(4)    C(3)    Y
C(5)    C(10)   X   C(5)    C(6)    Y
C(6)    C(5)    X   C(6)    C(7)    Y
C(7)    C(8)    X   C(7)    C(6)    Y
C(8)    C(9)    X   C(8)    N(2)    Y
C(9)    C(10)   X   C(9)    C(1)    Y
C(10)   C(9)    X   C(10)   C(5)    Y
C(11)   N(1)    X   C(11)   H(111)  Y
C(14)   N(2)    X   C(14)   H(141)  Y
CL(1A)  C(2A)   X   CL(1A)  CL(2A)  Y
CL(2A)  C(3A)   X   CL(2A)  CL(1A)  Y
O(1A)   C(1A)   X   O(1A)   H(1OO)  Y
O(2A)   C(1A)   X   O(2A)   CL(1A)  Y
O(3A)   C(4A)   X   O(3A)   CL(2A)  Y
O(4A)   C(4A)   X   O(4A)   H(1OO)  Y
C(1A)   O(1A)   X   C(1A)   O(2A)   Y
C(2A)   C(1A)   X   C(2A)   CL(1A)  Y
C(3A)   C(2A)   X   C(3A)   CL(2A)  Y
C(4A)   O(3A)   X   C(4A)   O(4A)   Y
H(1NN)  N(1)    X   H(1NN)  N(2)    Y
H(1OO)  O(1A)   X   H(1OO)  O(4A)   Y
H(111)  C(11)   X   H(111)  H(112)  Y
H(112)  C(11)   X   H(112)  H(111)  Y
H(113)  C(11)   X   H(113)  H(111)  Y
H(141)  C(14)   X   H(141)  H(142)  Y
H(142)  C(14)   X   H(142)  H(141)  Y
H(143)  C(14)   X   H(143)  H(141)  Y
H(2)    C(2)    X   H(2)    H(3)    Y
H(3)    C(3)    X   H(3)    H(2)    Y
H(4)    C(4)    X   H(4)    H(3)    Y
H(5)    C(5)    X   H(5)    H(4)    Y
H(6)    C(6)    X   H(6)    H(7)    Y
H(7)    C(7)    X   H(7)    H(6)    Y
```

The ATOM_LOCAL_AXES category is used to define a local Cartesian axis system for each atom. Data items in this category are looped, with the category key (`_atom_local_axes_atom_label`) being a pointer to the `_atom_site_label` defined in the ATOM_SITE category of the core CIF dictionary. The orientation of the Cartesian basis is defined in terms of vectors between atoms, and any arbitrary orientation can be described by including dummy atoms if necessary. The spherical coordinate system used in the description of the multipoles is defined in terms of the local Cartesian basis: z is the polar axis from which the angle θ is measured, φ is measured from the x axis in the xy plane with the y axis having a value of $\varphi = +90°$.

Example 3.5.3.1 shows how the category is used in part of an electron density CIF relating to a charge density study of the complex of 1,8-bis(dimethylamino)naphthalene with 1,2-dichloromaleic acid (a 'proton sponge') (Mallinson *et al.*, 1997).

3.5.3.2. Atom-centred multipole expansion functions

Data items in this category are as follows:

ATOM_RHO_MULTIPOLE

- `_atom_rho_multipole_atom_label`
  → `_atom_site_label`
  `_atom_rho_multipole_coeff_Pc`
  `_atom_rho_multipole_coeff_Pv`
  `_atom_rho_multipole_coeff_P00`
  `_atom_rho_multipole_coeff_P1-1`
  `_atom_rho_multipole_coeff_P10`
  `_atom_rho_multipole_coeff_P11`
  `_atom_rho_multipole_coeff_P2-2`
  `_atom_rho_multipole_coeff_P2-1`
  `_atom_rho_multipole_coeff_P20`
  `_atom_rho_multipole_coeff_P21`
  `_atom_rho_multipole_coeff_P22`
  `_atom_rho_multipole_coeff_P3-3`
  `_atom_rho_multipole_coeff_P3-2`
  `_atom_rho_multipole_coeff_P3-1`
  `_atom_rho_multipole_coeff_P30`
  `_atom_rho_multipole_coeff_P31`
  `_atom_rho_multipole_coeff_P32`
  `_atom_rho_multipole_coeff_P33`
  `_atom_rho_multipole_coeff_P4-4`
  `_atom_rho_multipole_coeff_P4-3`
  `_atom_rho_multipole_coeff_P4-2`
  `_atom_rho_multipole_coeff_P4-1`
  `_atom_rho_multipole_coeff_P40`
  `_atom_rho_multipole_coeff_P41`
  `_atom_rho_multipole_coeff_P42`
  `_atom_rho_multipole_coeff_P43`
  `_atom_rho_multipole_coeff_P44`
  `_atom_rho_multipole_configuration`
  `_atom_rho_multipole_core_source`
  `_atom_rho_multipole_kappa`
  `_atom_rho_multipole_kappa_prime0`
  `_atom_rho_multipole_kappa_prime1`
  `_atom_rho_multipole_kappa_prime2`
  `_atom_rho_multipole_kappa_prime3`
  `_atom_rho_multipole_kappa_prime4`
  `_atom_rho_multipole_radial_function_type`
  `_atom_rho_multipole_radial_slater_n0`
  `_atom_rho_multipole_radial_slater_n1`
  `_atom_rho_multipole_radial_slater_n2`
  `_atom_rho_multipole_radial_slater_n3`
  `_atom_rho_multipole_radial_slater_n4`
  `_atom_rho_multipole_radial_slater_zeta0`
  `_atom_rho_multipole_radial_slater_zeta1`
  `_atom_rho_multipole_radial_slater_zeta2`
  `_atom_rho_multipole_radial_slater_zeta3`
  `_atom_rho_multipole_radial_slater_zeta4`
  `_atom_rho_multipole_scat_core`
  `_atom_rho_multipole_scat_valence`
  `_atom_rho_multipole_valence_source`

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

Data items in this category are also looped. The category key (`_atom_rho_multipole_atom_label`) is also a child of `_atom_site_label` in the ATOM_SITE category, thereby linking the electron density multipole coefficients to a specific atom site in the list of atom coordinates. This category contains all the parameters of the Hansen & Coppens (1978) function

$$\rho(\mathbf{r}) = P_c\rho_{\text{core}}(\mathbf{r}) + P_v\kappa^3\rho_{\text{valence}}(\kappa r) + \sum_{0\le l\le l_{\max}}\{\kappa'(l)^3 R(\kappa'(l),l,\mathbf{r})\}\sum_{-l\le m\le l}\{P(l,m)d(l,m,\theta,\varphi)\},$$

where $\mathbf{r}$ is a position vector with respect to the atomic nucleus, and $\rho_{\text{core}}(\mathbf{r})$ and $\rho_{\text{valence}}(\kappa r)$ are the spherical core and valence electron densities, respectively. They are obtained from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974). They are also the Fourier transforms of the X-ray scattering factors given in `_atom_rho_multipole_scat_core` and `_atom_rho_multipole_scat_valence`.

P_c is the weight applied to the free-atom core (`_atom_rho_multipole_coeff_Pc`), P_v is the weight applied to the free-atom valence shell (`_atom_rho_multipole_coeff_Pv`), $P(0,0)$ is the number of remaining electrons (`_atom_rho_multipole_coeff_P00`) and $P_c + P_v + P(0,0) = Z$ (the atomic number) for

Example 3.5.3.2. *Multipole expansion for an atom in the proton sponge complex of Example 3.5.3.1.*

```
loop_
_atom_rho_multipole_atom_label
_atom_rho_multipole_coeff_Pv
_atom_rho_multipole_coeff_P00
_atom_rho_multipole_coeff_P11
_atom_rho_multipole_coeff_P1-1
_atom_rho_multipole_coeff_P10
_atom_rho_multipole_coeff_P20
_atom_rho_multipole_coeff_P21
_atom_rho_multipole_coeff_P2-1
_atom_rho_multipole_coeff_P22
_atom_rho_multipole_coeff_P2-2
_atom_rho_multipole_coeff_P30
_atom_rho_multipole_coeff_P31
_atom_rho_multipole_coeff_P3-1
_atom_rho_multipole_coeff_P32
_atom_rho_multipole_coeff_P3-2
_atom_rho_multipole_coeff_P33
_atom_rho_multipole_coeff_P3-3
_atom_rho_multipole_coeff_P40
_atom_rho_multipole_coeff_P41
_atom_rho_multipole_coeff_P4-1
_atom_rho_multipole_coeff_P42
_atom_rho_multipole_coeff_P4-2
_atom_rho_multipole_coeff_P43
_atom_rho_multipole_coeff_P4-3
_atom_rho_multipole_coeff_P44
_atom_rho_multipole_coeff_P4-4
_atom_rho_multipole_kappa
_atom_rho_multipole_kappa_prime0
_atom_rho_multipole_kappa_prime1
_atom_rho_multipole_kappa_prime2
_atom_rho_multipole_kappa_prime3
_atom_rho_multipole_kappa_prime4
_atom_rho_multipole_configuration
_atom_rho_multipole_radial_slater_n0
_atom_rho_multipole_radial_slater_zeta0
_atom_rho_multipole_radial_slater_n1
_atom_rho_multipole_radial_slater_zeta1
_atom_rho_multipole_radial_slater_n2
_atom_rho_multipole_radial_slater_zeta2
_atom_rho_multipole_radial_slater_n3
_atom_rho_multipole_radial_slater_zeta3
_atom_rho_multipole_radial_slater_n4
_atom_rho_multipole_radial_slater_zeta4
_atom_rho_multipole_core_source
_atom_rho_multipole_valence_source
N(1)    2.63(5)
  0.00       -0.037(17)  0.062(14)  0.00     -0.084(18)
  0.00        0.00       -0.027(15) -0.048(13)  0.00
 -0.098(16) -0.063(14)   0.00        0.00      0.082(14)
 -0.037(14)  0.00        0.00        0.00      0.00
  0.00       0.00        0.00        0.00      0.00
0.992(8)     0.80(4)     0.80     0.80     0.80    0.80
;
1S 2S 3S 4S 2P 3P 4P 3D 4D 4F 5S 5P 6S 6P 5D 7S 6D 5F
 2 -2  0  0 -3  0  0  0  0  0  0  0  0  0  0  0  0  0
;
   2   7.2553  2   7.2553  2   7.2553  3   7.2553
   4   7.2553
'Clementi & Roetti, 1974'
'Clementi & Roetti, 1974'

# Data for other atoms have been omitted for brevity
```

a neutral atom. κ is the valence electron expansion factor (`_atom_rho_multipole_kappa`), $R(\kappa'(l), l, \mathbf{r})$ is the radial function (Slater or equivalent) (`_atom_rho_multipole_radial_*`), $\kappa'(l)$ is the multipole function expansion factor (`_atom_rho_multipole_kappa_prime[l]`), $P(l,m)$ are the spherical harmonic coefficients (`_atom_rho_multiple_coeff_P[lm]`) and $d(l,m,\theta,\varphi)$ is the spherical harmonic of order l,m at the angle (θ,φ). The summations are performed over the index ranges $0 \leq l \leq l_{\max}$, $-l \leq m \leq l$, where $l_{\max}$ is the highest order of multipole applied.

Example 3.5.3.2 demonstrates how the category is used in the proton sponge complex of Example 3.5.3.1. Only the first atom is shown in the example.

3.5.4. Development of the dictionary and supporting software

The first implementation of the dictionary appeared in a version of the *XD* program package (Koritsanszky *et al.*, 2003). Thus *XD* can read and write CIFs which include data items in the two categories described above (as well as, of course, items in the core CIF dictionary). It is envisaged that future developments to the rhoCIF dictionary will add features that are relevant to other widely used program packages. In the *XD* implementation, use was made of the *CIFtbx* library of Fortran functions for programming CIF applications (Hall & Bernstein, 1996).

We thank the users of the program package *XD* (Koritsanszky *et al.*, 2003) who provided feedback during the development of the rhoCIF dictionary and Brian McMahon for helpful comments on its presentation.

References

Bader, R. F. W. (1990). *Atoms in molecules: a quantum theory.* Oxford University Press.

Clementi, E. & Roetti, C. (1974). *Roothan–Hartree–Fock atomic wavefunctions. Basis functions and their coefficients for ground and certain excited states of neutral and ionized atoms. At. Data Nucl. Data Tables*, **14**, 177–478.

Coppens, P. (1997). *X-ray charge densities and chemical bonding.* Oxford University Press.

Hall, S. R. & Bernstein, H. J. (1996). *CIF applications. V. CIFtbx2: extended tool box for manipulating CIFs. J. Appl. Cryst.* **29**, 598–603.

Hansen, N. K. & Coppens, P. (1978). *Testing aspherical atom refinements on small-molecule data sets. Acta Cryst.* A**34**, 909–921.

Hohenberg, P. & Kohn, W. (1964). *Inhomogeneous electron gas. Phys. Rev. (Second Series)*, **136**, B864–B871.

Kohn, W., Becke, A. D. & Parr, R. G. (1996). *Density functional theory of electronic structure. J. Phys. Chem.* **100**, 12974–12980.

Koritsanszky, T., Richter, T., Macchi, P., Volkov, A., Gatti, C., Howard, S., Mallinson, P. R., Farrugia, L., Su, Z. & Hansen, N. K. (2003). *XD. Computer program package for multipole refinement and topological analysis of electron densities from diffraction data.* http://xd.chem.buffalo.edu.

Koritsanszky, T. S. & Coppens, P. (2001). *Chemical applications of X-ray charge density analysis. Chem. Rev.* **101**, 1583–1627.

Labanowski, J. K. & Andzelm, J. W. (1991). *Density-functional methods in chemistry.* New York: Springer-Verlag.

Mallinson, P. R., Wozniak, K., Smith, G. T. & McCormack, K. L. (1997). *A charge density analysis of cationic and anionic hydrogen bonds in a 'proton sponge' complex. J. Am. Chem. Soc.* **119**, 11502–11509.

Nagy, A. (1998). *Density functional. Theory and application to atoms and molecules. Phys. Rep.* **298**, 2–79.

Spackman, M. A. & Brown, A. S. (1994). *Charge densities from X-ray diffraction data. R. Soc. Chem. Ann. Rep. Sect. C*, pp. 175–212.

Stewart, R. F. (1973). *Electron population analysis with generalized X-ray scattering factors: higher multipoles. J. Chem. Phys.* **58**, 1668–1676.

Ziegler, T. (1991). *Approximate density functional theory as a practical tool in molecular energetics and dynamics. Chem. Rev.* **91**, 651–667.

3.6. Classification and use of macromolecular data

BY P. M. D. FITZGERALD, J. D. WESTBROOK, P. E. BOURNE, B. MCMAHON, K. D. WATENPAUGH AND H. M. BERMAN
WITH APPENDIX 3.6.2 BY J. D. WESTBROOK, K. HENRICK, E. L. ULRICH AND H. M. BERMAN

3.6.1. Introduction

As described in Chapter 1.1, the macromolecular crystallographic information file (mmCIF) dictionary (Fitzgerald *et al.*, 1996; Bourne *et al.*, 1997) was initially commissioned as an extension to the core CIF dictionary (Hall *et al.*, 1991), with the intention of adding data names suitable for a full description of a macromolecular crystallographic experiment and its results. However, the need to specify relationships between the data items describing different components of a complex macromolecular structure led to the development of a richer dictionary definition language (DDL2). The data names were then defined according to the DDL2 formalism. For consistency, the existing core dictionary data items were also recast in the DDL2 formalism. Since no other DDL2 applications were envisaged at that time, the core items were then embedded in the mmCIF dictionary as a subset of the complete dictionary. The current release of the mmCIF dictionary described in this chapter includes all the data items in version 2.3.1 of the core dictionary. The mmCIF dictionary is not routinely updated to match additions to the core dictionary, but it is expected that when new versions of the mmCIF dictionary are released to meet the requirements of the macromolecular community, the most recent version of the core dictionary will be incorporated in the new mmCIF dictionary as part of the revision.

The resulting stand-alone dictionary is very large and is described in detail in this chapter. The philosophy behind the design of the dictionary is discussed in Section 3.6.2 and an example of its use is given and discussed in Section 3.2.3. The contents of the dictionary are then described in the remainder of the chapter, starting at Section 3.6.4. The discussion follows the sequence of Table 3.1.10.1: experimental measurements, analysis, structure, publication and file metadata are considered in turn. The discussion of individual categories may be found by using the overview of the dictionary structure given in Appendix 3.6.1.

The data names in the mmCIF dictionary derived from the core CIF dictionary differ from their DDL1 counterparts in that a full stop (.) is used to designate explicitly the category to which the data name belongs, *e.g.* `_cell.length_a` is used in place of `_cell_length_a`. Sometimes the mmCIF counterpart of a core data name may have a different form, for example to enforce the rule in DDL2 that the category name is the initial part of any data name within that category. This convention is generally observed in DDL1, but is not mandatory. Formally, the corresponding DDL1 core data name is obtained from the `_item_aliases.alias_name` attribute of the definition. The provision of a formal alias for all data names derived from the core dictionary allows a DDL2-compliant parser to read and interpret a data file constructed according to the DDL1 dictionary described in Chapter 3.2. Achieving this compatibility with CIFs built using DDL1 dictionaries was a very important goal in the design of DDL2 and the mmCIF dictionary.

In this chapter, categories and individual data names that correspond to matching entries in the core dictionary are not discussed in detail unless they are used in a different way in mmCIF. Chapter 3.2 should therefore be read first for a description of the categories common to both the core and mmCIF dictionaries. This chapter concentrates on the categories specific to mmCIF. Formal differences between mmCIF categories and core CIF categories are also summarized.

3.6.2. Considerations underlying the design of the dictionary

From the outset, mmCIF was envisaged as a providing a more detailed description of macromolecular structures than the existing Protein Data Bank (PDB) format (Chapter 1.1). A number of considerations guided the development of version 1 of the mmCIF dictionary. These included:

(i) Every field of every PDB record type should be represented by an mmCIF data item if the PDB field is important for describing the structure, the experiment that was conducted in determining the structure or the revision history of the entry. It is important to note that it is straightforward to convert an mmCIF data file to a PDB file without loss of information, since all the information is parsable. It is not possible, however, to automate completely the conversion of a PDB file to an mmCIF, since many mmCIF data items are either not present in the PDB file or are present in PDB REMARK records that in some cases cannot be parsed. The contents of PDB REMARK records are maintained as separate data items within mmCIF so as to preserve all the information, even if the information is not parsable.

(ii) Data items should be defined so that all the information given in the materials and methods section of an article describing the structure can be referenced. This includes major features of the crystal, the diffraction experiment, the phasing calculations and the refinement.

(iii) Data items should be provided for describing the biologically active molecule and any important structural subcomponents.

(iv) It should be possible to represent atom positions using either orthogonal ångström or fractional coordinates.

(v) Data items should be provided for describing the initial experimental reflection data, including all the data sets used in the phasing of the structure, and the final processed data.

(vi) Crystallographic and noncrystallographic symmetry should be described.

Affiliations: PAULA M. D. FITZGERALD, Merck Research Laboratories, Rahway, New Jersey, USA; JOHN D. WESTBROOK, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, New Jersey, USA; PHILLIP E. BOURNE, Research Collaboratory for Structural Bioinformatics, San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0537, USA; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England; KEITH D. WATENPAUGH, retired; formerly Structural, Analytical and Medicinal Chemistry, Pharmacia Corporation, Kalamazoo, Michigan, USA; HELEN M. BERMAN, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, New Jersey, USA; KIM HENRICK, EMBL Outstation, The European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, England; ELDON L. ULRICH, Department of Biochemistry, University of Wisconsin Madison, 433 Babcock Drive, Madison, WI 53706-1544, USA.

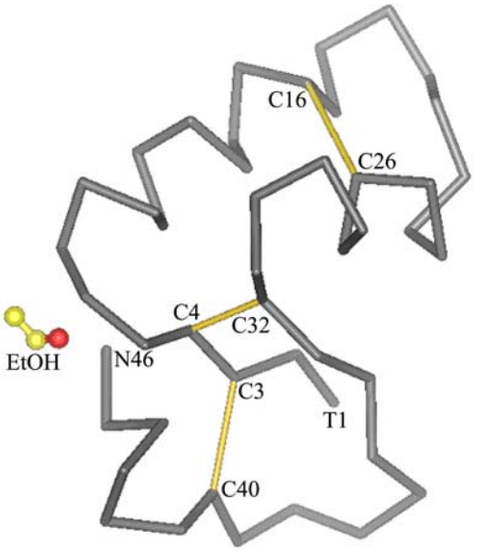


Fig. 3.6.3.1. A representation of crambin (PDB 3CNR) with a co-crystallized ethanol molecule.

(vii) Data items should be present for describing the characteristics and geometry of canonical and non-canonical amino acids, nucleotides, sugars and ligand groups.

(viii) Data items should be provided that permit a detailed description of the chemistry of the component parts of the macromolecule to be given.

(ix) Data items should be present that provide specific pointers from elements of the structure (*e.g.* the sequence, bound inhibitors) to appropriate entries in publicly available databases.

(x) Data items should be present that provide meaningful three-dimensional views of the structure so as to highlight functional and structural aspects of the macromolecule.

(xi) Data items specific to an NMR experiment or modelling study would not in general be included in version 1. However, data items that summarize the features of an ensemble of structures and permit a description of each member of the ensemble to be given should be available.

(xii) A comprehensive set of data items for providing a higher-order structure description (for example, to cover supersecondary structure and functional classification) was considered to be beyond the scope of version 1.

Based on the above, the first version of the mmCIF dictionary with approximately 1700 data items (including those data items taken from the core CIF dictionary) was developed and officially approved in October 1997. Subsequent revisions have increased the number of data items to over 2000. It is not expected that all the data items will be present in every mmCIF data file. Instead, the goal was to provide a wide range of data items from which users can select those that best suit the structure they wish to describe.

3.6.3. Overview of the mmCIF data model

The solution and refinement of a macromolecular structure is complex and often difficult, as there are a large number of atoms in a typical macromolecule, the molecular conformation can be complex and it can be difficult to model included solvent molecules. However, even when a satisfactory structural model has been derived, describing the structure can be a considerable challenge. Using diagrams can help, but two-dimensional projections are often inadequate for illustrating important features and a complete understanding of the three-dimensional structure of a macromolecule can often only be reached by using interactive molecular graphics software.

Example 3.6.3.1. *Specification of the three distinct components of the crambin structure.*

```
loop_
  _struct_asym.id
  _struct_asym.entity_id
  _struct_asym.details
   chain_a A       'single polypeptide chain'
   ethanol ethanol 'cocrystallized ethanol molecule'
   water   HOH     .
```

The mmCIF dictionary provides several ways for describing the structure. The PUBL categories can be used to record text describing the structure. The complete list of atomic coordinates may be used as input for visualization programs that allow a range of wire-frame, stick, space-filling, ribbon or cartoon representations to be generated based upon inbuilt heuristics and user interaction. However, most importantly, the mmCIF approach also offers a large collection of categories which are designed to provide descriptions of the structure at different levels of detail, and the relationships between data items in different categories permit the function of an individual atom site at any particular level of detail to be traced.

Before beginning the detailed description of the full mmCIF dictionary, it is helpful to demonstrate how it is used to describe the structure of a biological macromolecule. Fig. 3.6.3.1 shows the small protein crambin, which is a single polypeptide chain of 48 residues. The molecule co-crystallizes with a molecule of ethanol, although this is not thought to have any biological effect. Almost a quarter of the residues have side chains that adopt alternative conformations, and there is sequence heterogeneity at positions 22 (Pro/Ser) and 25 (Leu/Ile). Three disulfide links stabilize the structure.

The highest level of the description of the structure uses data items from the STRUCT category group. The crystallographic asymmetric unit contains one protein molecule, one co-crystallization ethanol molecule and a water solvent molecule. These are described with data items from the STRUCT_ASYM category (Example 3.6.3.1).

Each entry in this list assigns a label to a discrete component of the asymmetric unit and associates it with an entry in the entity list that defines each distinct chemical species in the crystal (Example 3.6.3.2).

The biological functions of the components of the crystal structure are described using data items in the STRUCT_BIOL and related categories. For crambin, the biological function is still unknown (see Example 3.6.3.3). This example also shows how the biological unit is generated from specific discrete objects in the asymmetric unit. In this case the relationship is trivial, but it will often be much more complex.

The secondary structure of the protein is described using data items in the STRUCT_CONF category (and in the STRUCT_SHEET category where relevant). The beginning and end labels for each

Example 3.6.3.2. *Specification of the distinct chemical entities in the crambin structure.*

```
loop_
  _entity.id
  _entity.type
  _entity.formula_weight
  _entity.src_method
    A        polymer      4716     natural
    ethanol  non-polymer    52     synthetic
    HOH      water          18     .
```

Example 3.6.3.3. *Identification of the biological function of the components of the crambin structure.*

```
_struct_biol.id                 crambin_1
_struct_biol.details
; The function of this protein is unknown and
  therefore the biological unit is assumed to be
  the single polypeptide chain without
  co-crystallization factors i.e. ethanol.
;
_struct_biol_gen.biol_id        crambin_1
_struct_biol_gen.asym_id        chain_a
_struct_biol_gen.symmetry       1_555
```

α-helix, β-strand or turn in Example 3.6.3.4 refer to the chemical components of the structural unit labelled chain_a at the given locations in the sequence (*e.g.* helix H1 runs from the isoleucine at position number 7 to the proline at position number 19 in the amino-acid sequence).

Interactions between different parts of the structure are described using data items in the STRUCT_CONN and related categories. In Example 3.6.3.5, some of the disulfide bridges and intramolecular hydrogen bonds are reported. As with the secondary structural elements, the partners in the links are identified by complex labels that include the chemical component involved, the object within the asymmetric unit that is under consideration, the position in the amino-acid (or nucleotide) sequence and the individual atom.

The objects identified at the highest level of the description of the structure are arbitrary. To discover their chemical identity, one needs to consult the ENTITY category group. As indicated above, each separate chemical species in the crystal should be specified in the entity table. Chemical entities are classified as polymer, non-polymer or water. Non-polymeric molecules, such as the co-crystallized ethanol in this example, are described as distinct chemical components using data items in the CHEM_COMP family of categories. Polymeric molecules are described using data items in the ENTITY_POLY family of categories.

In Example 3.6.3.6, the natural source for crambin is described, the overall features of the polypeptide chain are listed and the component parts (in effect the amino-acid sequence) are tabulated. Note that sequence heterogeneity is described by allowing a sequence number to be correlated with more than one monomer identifier (in the example, sequence number 22 is assigned both to proline and serine, while 25 is assigned to both leucine and isoleucine). Sequence heterogeneity can be defined by assigning suitable labels in the ATOM_SITE list.

The individual amino acids in the protein sequence of Example 3.6.3.6 are labelled by the data item `_entity_poly_seq.mon_id`; this refers to the separate chemical components listed in the CHEM_COMP family of categories (Example 3.6.3.7). As mentioned above, entries in these categories may be individual monomeric species within the crystal structure, or they may be amino acids or nucleotide bases that form the macromolecular polymer. In most cases, the entries recorded in these categories will be summaries of chemical information for standard amino acids and nucleotides, or references to external libraries of standard data for these. However, the categories contain enough data items to describe modified residues or co-crystallization factors in full if necessary.

At the most detailed level, the individual atom sites are described with data items in the ATOM category group, as shown for crambin in Example 3.6.3.8. A few points about this

Example 3.6.3.4. *Description of the secondary structure of crambin.*

```
loop_
  _struct_conf.id
  _struct_conf.conf_type_id
  _struct_conf.beg_label_comp_id
  _struct_conf.beg_label_asym_id
  _struct_conf.beg_label_seq_id
  _struct_conf.end_label_comp_id
  _struct_conf.end_label_asym_id
  _struct_conf.end_label_seq_id
  _struct_conf.details
    H1 HELX_RH_AL_P ILE chain_a  7 PRO chain_a 19
                                 'HELX-RH3T 17-19'
    H2 HELX_RH_AL_P GLU chain_a 23 THR chain_a 30
                                'Alpha-N start'
    S1 STRN_P       CYS chain_a 32 ILE chain_a 35 .
    S2 STRN_P       THR chain_a  1 CYS chain_a  4 .
    S3 STRN_P       ASN chain_a 46 ASN chain_a 46 .
    S4 STRN_P       THR chain_a 39 PRO chain_a 41 .
    T1 TURN-TY1_P   ARG chain_a 17 GLY chain_a 20 .
    T2 TURN-TY1_P   PRO chain_a 41 TYR chain_a 44 .
```

Example 3.6.3.5. *Interactions between parts of the crambin structure.*

```
loop_
  _struct_conn.id
  _struct_conn.conn_type_id
  _struct_conn.ptnr1_label_comp_id
  _struct_conn.ptnr1_label_asym_id
  _struct_conn.ptnr1_label_seq_id
  _struct_conn.ptnr1_label_atom_id
  _struct_conn.ptnr1_role
  _struct_conn.ptnr1_symmetry
  _struct_conn.ptnr2_label_comp_id
  _struct_conn.ptnr2_label_asym_id
  _struct_conn.ptnr2_label_seq_id
  _struct_conn.ptnr2_label_atom_id
  _struct_conn.ptnr2_role
  _struct_conn.ptnr2_symmetry
  _struct_conn.details
    SS1 disulf CYS chain_a  3 S  .        1_555
               CYS chain_a 40 S  .        1_555 .
    SS2 disulf CYS chain_a  4 S  .        1_555
               CYS chain_a 32 S  .        1_555 .
    HB1 hydrog SER chain_a  6 OG positive 1_555
               LEU chain_a  8 O  negative 1_556 .
    HB2 hydrog ARG chain_a 17 N  positive 1_555
               ASP chain_a 43 O  negative 1_554 .
```

Example 3.6.3.6. *Description of the crambin polypeptide.*

```
_entity_name_com.entity_id      A
_entity_name_com.name           crambin
_entity_src_nat.entity_id       A
_entity_src_nat.common_name     'Abyssinian Cabbage'
_entity_src_nat.genus           Crambe
_entity_src_nat.species         abyssinica
_entity_src_nat.details         ?
_entity_poly.entity_id          A
_entity_poly.type               polypeptide(L)
_entity_poly.nstd_chirality     no
_entity_poly.nstd_linkage       no
_entity_poly.nstd_monomer       no
_entity_poly.type_details
 'Sequence heterogeneity at residues 22 and 25'
loop_
  _entity_poly_seq.entity_id
  _entity_poly_seq.num
  _entity_poly_seq.mon_id
    A    1   THR     A    2   THR
# - - abbreviated - - -
    A   22   PRO     A   22   SER
    A   23   GLU     A   24   ALA
    A   25   LEU     A   25   ILE
# - - abbreviated - - -
    A   47   ALA     A   48   ASN
```

Example 3.6.3.7. *Separate chemical components forming the crambin polypeptide.*

```
loop_
_chem_comp.id
_chem_comp.mon_nstd_flag
_chem_comp.formula
_chem_comp.name
  ethanol .   'C2 H6 O1'         "ethanol"
  ALA     yes 'C3 H7 N1 O2'      "alanine"
  ARG     yes 'C6 H14 N4 O2'     "arginine"
  ASN     yes 'C4 H8 N2 O3'      "asparagine"
  ASP     yes 'C4 H7 N1 O4'      "aspartic acid"
  CYS     yes 'C3 H7 N1 O2 S1'   "cysteine"
  GLU     yes 'C5 H9 N1 O4'      "glutamic acid"
  GLY     yes 'C2 H5 N1 O2'      "glycine"
  ILE     yes 'C6 H13 N1 O2'     "isoleucine"
  LEU     yes 'C6 H13 N1 O2'     "leucine"
  PHE     yes 'C9 H11 N1 O2'     "phenylalanine"
  PRO     yes 'C5 H9 N1 O2'      "proline"
  SER     yes 'C3 H7 N1 O3'      "serine"
  THR     yes 'C4 H9 N1 O3'      "threonine"
  TYR     yes 'C9 H11 N1 O3'     "tyrosine"
  VAL     yes 'C5 H11 N1 O2'     "valine"
```

example should be noted. The composite labelling of each site includes a pointer to the description of the parent molecule as a specific object in the asymmetric unit (`_atom_site.label_asym_id`) and to the relevant monomeric building block of which the atom is a member (`_atom_site.label_comp_id`). The label component `_atom_site.label_alt_id` indicates alternative conformations in which an atom site may be found. For example, the atom sites numbered 3 and 4 are alternative locations for the α-carbon of the terminal residue. It may be deduced from the occupancies that the alternative conformations A and B are modelled with 80% and 20% occupancy, respectively, but this can be stated explicitly using the ATOM_SITES_ALT category. The sequence heterogeneity at residue 22 is shown by the presence of pointers to proline and serine, and the occupancy factors show that proline and serine are present in the ratio 60 to 40. There is also an alternative conformation within the serine at residue 22, split equally across two sites.

Example 3.6.3.8. *Partial listing of the atomic coordinates of crambin.*

```
loop_
_atom_site.label_seq_id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_alt_id
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.footnote_id
_atom_site.label_entity_id
_atom_site.id
1  N  N    THR chain_a A  16.864  14.059   3.442
   0.80  6.22  .   A    1
1  N  N    THR chain_a B  17.633  14.126   4.146
   0.20  8.40  .   A    2
1  C  CA   THR chain_a A  16.868  12.814   4.233
   0.80  4.45  .   A    3
1  C  CA   THR chain_a B  17.282  12.671   4.355
   0.20  7.82  .   A    4
1  C  C    THR chain_a .  15.583  12.775   4.990
   1.00  4.39  .   A    5
1  O  O    THR chain_a .  15.112  13.824   5.431
   1.00  7.04  .   A    6
1  C  CB   THR chain_a A  18.060  12.807   5.200
   0.80  5.42  .   A    7
1  C  CB   THR chain_a B  18.202  11.709   5.108
   0.20 11.07  .   A    8
1  O  OG1  THR chain_a A  19.233  12.892   4.380
   0.80  7.87  .   A    9
1  O  OG1  THR chain_a B  17.662  10.381   4.831
   0.20 14.39  .   A   10
1  C  CG2  THR chain_a A  18.117  11.578   6.092
   0.80  6.88  .   A   11
1  C  CG2  THR chain_a B  17.973  11.955   6.599
   0.20 19.74  .   A   12
# - - abbreviated - - -
22 N  N    PRO chain_a .   4.909  12.659  -3.127
   0.60  3.03  .   A  352
22 C  CA   PRO chain_a .   6.035  13.459  -2.622
   0.60  3.04  .   A  353
22 C  C    PRO chain_a .   6.362  13.139  -1.174
   0.60  3.08  .   A  354
22 O  O    PRO chain_a .   5.473  12.959  -0.323
   0.60  3.67  .   A  355
22 C  CB   PRO chain_a .   5.528  14.895  -2.825
   0.60  4.19  .   A  356
22 C  CG   PRO chain_a .   4.614  14.846  -4.059
   0.60  3.91  .   A  357
22 C  CD   PRO chain_a .   3.904  13.493  -3.885
   0.60  3.25  .   A  358
22 N  N    SER chain_a .   4.909  12.659  -3.127
   0.40  3.03  .   A  366
22 C  CA   SER chain_a .   6.035  13.459  -2.622
   0.40  3.04  .   A  367
22 C  C    SER chain_a .   6.362  13.139  -1.174
   0.40  3.08  .   A  368
22 O  O    SER chain_a .   5.473  12.959  -0.323
   0.40  3.67  .   A  369
22 C  CB   SER chain_a .   5.644  14.934  -2.679
   0.40  3.96  .   A  370
22 O  OG   SER chain_a C   4.712  15.250  -1.677
   0.20  3.53  .   A  371
22 O  OG   SER chain_a D   6.688  15.800  -2.315
   0.20  7.09  .   A  372
```

3.6.4. Content of the macromolecular CIF dictionary

Because it is derived from the core CIF dictionary, the mmCIF dictionary shares the same general structure as outlined in Chapter 3.2. However, DDL2 permits the formal assignment of categories to *category groups*. Table 3.6.4.1 lists the major category groups in the mmCIF dictionary (a full list is given in Appendix 3.6.1 and at the beginning of Chapter 4.5).

Small capitals are used for the names of category groups and individual categories in this volume, but the identifiers in the dictionary are actually lower-case strings.

The ordering of category groups in the remainder of this chapter follows the thematic scheme of Table 3.1.10.1. The discussion proceeds under the headings *Experimental measurements* (Section 3.6.5), *Analysis* (Section 3.6.6), *Atomicity, chemistry and structure* (Section 3.6.7), *Publication* (Section 3.6.8) and *File metadata* (Section 3.6.9).

Certain conventions of style and layout have been followed to summarize the large amount of information in the mmCIF dictionary and to help the reader navigate their way through this chapter. Appendix 3.6.1 is an overview of the mmCIF dictionary structure by category and lists all the categories with the number of the section in which they are discussed. This acts as an index between the alphabetical ordering within the dictionary and the thematic ordering of this chapter. Each thematic section lists the categories discussed in that section. Within each subsection, the data names within the relevant categories are listed. Category keys, pointers to parent data items and aliases to data items in the core CIF dictionary are indicated. For each category, the data item (or set of data items that must be considered together) that forms the category key is marked by a bullet (•) and listed first; the other data names follow in alphabetical order.

For measured or derived numerical quantities that should be specified with a standard uncertainty (in older terminology, an estimated standard deviation), the core dictionary uses the DDL1

Table 3.6.4.1. *Major category groups defined in the mmCIF dictionary*

The groups are listed in the order in which they are described in this chapter. There is also an INCLUSIVE category group, which serves as a formal higher-order container group to which all other category groups belong.

Section	Category group	Subject covered
(*a*) Experimental measurements		
3.6.5.1	CELL	Unit cell
3.6.5.2	DIFFRN	Diffraction experiment
3.6.5.3	EXPTL	Experimental conditions
(*b*) Analysis		
3.6.6.1	PHASING	Phasing techniques
3.6.6.2	REFINE	Refinement procedures
3.6.6.3	REFLN	Reflection measurements
(*c*) Atomicity, chemistry and structure		
3.6.7.1	ATOM	Atom sites
3.6.7.2	CHEMICAL	Chemical properties and nomenclature
3.6.7.3	ENTITY	Chemical entities
3.6.7.4	GEOM	Geometry of atom sites
3.6.7.5	STRUCT	Crystallographic structure
3.6.7.6	SYMMETRY	Symmetry information
3.6.7.7	VALENCE	Bond-valence information
(*d*) Publication		
3.6.8.1	CITATION	Bibliographic references
3.6.8.2	COMPUTING	Computational details of the experiment
3.6.8.3	DATABASE	Database information
3.6.8.4	IUCR	Journal housekeeping and the contents of a published article
(*e*) File metadata		
3.6.9.1	AUDIT	Dictionary maintenance and identification
3.6.9.2	ENTRY	Links between data blocks
3.6.9.3	COMPLIANCE	Compliance with previous dictionaries

attribute `_type_conditions_esd` and allows the standard uncertainty of the value to be placed in parentheses after the numerical value, as in

```
_cell_length_a      58.39(5)
```

This is also permitted in mmCIF, but it is preferable to use a separate data item to record the standard uncertainty, as in

```
_cell.length_a        58.39
_cell.length_a_esd     0.05
```

There are many of these kinds of data names in the mmCIF dictionary. The name of each is derived by adding `_esd` to the data name for the value. They are indicated by a + symbol in the category summaries in this chapter.

3.6.5. Experimental measurements

The CELL, DIFFRN and EXPTL category groups are used to describe the crystallographic experiment. The data items used for this purpose in mmCIF are for the most part identical to those in the core CIF dictionary. A complete discussion of the data names in each category may be found in Section 3.2.2.

mmCIF also contains the new categories EXPTL_CRYSTAL_GROW and EXPTL_CRYSTAL_GROW_COMP (Section 3.6.5.3.2), which are used to provide a more structured description of crystallization than is available in the core CIF dictionary.

3.6.5.1. Crystal cell parameters and measurement conditions

The categories describing the crystal unit cell and its determination are as follows:

CELL group
- CELL
- CELL_MEASUREMENT
- CELL_MEASUREMENT_REFLN

The mmCIF dictionary differs from the core CIF dictionary in assigning separate categories to data names that define the crystal unit-cell parameters and to data names relating to the experimental determination of the unit cell. Details of the unit-cell parameters are given in the CELL category and data items in the distinct CELL_MEASUREMENT category are used to describe how the unit-cell parameters were measured. The category CELL_MEASUREMENT_REFLN, which is used to list the reflections used in the unit-cell determination, is common to the core and mmCIF dictionaries.

The data items in these categories are as follows:

(*a*) CELL

```
• _cell.entry_id
        → _entry.id
+ _cell.angle_alpha
+ _cell.angle_beta
+ _cell.angle_gamma
  _cell.details (~ _cell_special_details)
  _cell.formula_units_Z
+ _cell.length_a
+ _cell.length_b
+ _cell.length_c
+ _cell.reciprocal_angle_alpha
+ _cell.reciprocal_angle_beta
+ _cell.reciprocal_angle_gamma
+ _cell.reciprocal_length_a
+ _cell.reciprocal_length_b
+ _cell.reciprocal_length_c
+ _cell.volume
  _cell.Z_PDB
```

(*b*) CELL_MEASUREMENT

```
• _cell_measurement.entry_id
        → _entry.id
+ _cell_measurement.pressure
  _cell_measurement.radiation
  _cell_measurement.reflns_used
+ _cell_measurement.temp
        (~ _cell_measurement_temperature)
  _cell_measurement.theta_max
  _cell_measurement.theta_min
  _cell_measurement.wavelength
```

(*c*) CELL_MEASUREMENT_REFLN

```
• _cell_measurement_refln.index_h
• _cell_measurement_refln.index_k
• _cell_measurement_refln.index_l
  _cell_measurement_refln.theta
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ~ symbol. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string `_esd` to the data name listed.

The summary above includes the formal category keys that have been introduced in mmCIF because the corresponding core categories do not expect looped data, and therefore do not require the specification of a unique identifier. In the relational model of DDL2, all categories are considered to be tables and therefore each category must have a unique identifier. Where core CIF categories have one or more data names that fulfil the role of table-row identifiers, these have generally been carried over as category keys in the mmCIF dictionary (for example, the data items that correspond to the *h*, *k*, and *l* Miller indices of a reflection in the CELL_MEASUREMENT_REFLN category).

Example 3.6.5.1. *Cell constants and their measurement for an HIV-1 protease crystal (PDB 5HVP) described with data items in the* CELL *and* CELL_MEASUREMENT *categories (Fitzgerald et al., 1990).*

```
_cell.entry_id                          '5HVP'
_cell.length_a                          58.39
_cell.length_a_esd                       0.05
_cell.length_b                          86.70
_cell.length_b_esd                       0.12
_cell.length_c                          46.27
_cell.length_c_esd                       0.06
_cell.angle_alpha                       90.00
_cell.angle_beta                        90.00
_cell.angle_gamma                       90.00
_cell.volume                           234237
_cell.details
; The cell parameters were refined every twenty
  frames during data integration. The cell lengths
  given are the mean of 55 such refinements; the
  esds given are the root-mean-square deviations
  of these 55 observations from that mean.
;
_cell_measurement.entry_id              '5HVP'
_cell_measurement.temp                    293
_cell_measurement.temp_esd                  3
_cell_measurement.theta_min                11
_cell_measurement.theta_max                31
_cell_measurement.wavelength             1.54
```

Example 3.6.5.1 shows how data items from these categories are used in practice and shows the use of separate data items to record standard uncertainties of measurable quantities.

3.6.5.2. Data collection

The categories describing data collection are as follows:

DIFFRN group
- DIFFRN
- DIFFRN_ATTENUATOR
- DIFFRN_DETECTOR
- DIFFRN_MEASUREMENT
- DIFFRN_ORIENT_MATRIX
- DIFFRN_ORIENT_REFLN
- DIFFRN_RADIATION
- DIFFRN_RADIATION_WAVELENGTH
- DIFFRN_REFLN
- DIFFRN_REFLNS
- DIFFRN_REFLNS_CLASS
- DIFFRN_SCALE
- DIFFRN_SOURCE
- DIFFRN_STANDARD_REFLN
- DIFFRN_STANDARDS

The categories in the DIFFRN category group describe the diffraction experiment. Data items in the DIFFRN category itself can be used to give overall information about the experiment, such as the temperature and pressure. Examples of the other categories are DIFFRN_DETECTOR, which is used for describing the detector used for data collection, and DIFFRN_SOURCE, which is used to give details of the source of the radiation used in the experiment. Data items in the DIFFRN_REFLN category can be used to give information about the raw data and data items in the DIFFRN_REFLNS category can be used to give information about all the reflection data collectively.

The data items in the categories in the DIFFRN group are as follows:

(*a*) DIFFRN
- **_diffrn.id**

  _diffrn.ambient_environment
+ **_diffrn.ambient_pressure**

  _diffrn.ambient_pressure_gt

  _diffrn.ambient_pressure_lt
+ **_diffrn.ambient_temp** (~ **_diffrn_ambient_temperature**)

  _diffrn.ambient_temp_details

  _diffrn.ambient_temp_gt

  _diffrn.ambient_temp_lt

  _diffrn.crystal_id (~ **_diffrn_refln_crystal_id**)

  _diffrn.crystal_support

  _diffrn.crystal_treatment

  _diffrn.details (~ **_diffrn_special_details**)

(*b*) DIFFRN_ATTENUATOR
- **_diffrn_attenuator.code**

  _diffrn_attenuator.material

  _diffrn_attenuator.scale

(*c*) DIFFRN_DETECTOR
- **_diffrn_detector.diffrn_id**
  → **_diffrn.id**

  _diffrn_detector.area_resol_mean

  _diffrn_detector.details

  _diffrn_detector.detector (~ **_diffrn_detector**)

  _diffrn_detector.dtime

  _diffrn_detector.type

(*d*) DIFFRN_MEASUREMENT
- **_diffrn_measurement.diffrn_id**
  → **_diffrn.id**

  _diffrn_measurement.details

  _diffrn_measurement.device

  _diffrn_measurement.device_details

  _diffrn_measurement.device_type

  _diffrn_measurement.method

  _diffrn_measurement.specimen_support

(*e*) DIFFRN_ORIENT_MATRIX
- **_diffrn_orient_matrix.diffrn_id**
  → **_diffrn.id**

  _diffrn_orient_matrix.type

  _diffrn_orient_matrix.UB[1][1]
  (~ **_diffrn_orient_matrix_UB_11**)

  _diffrn_orient_matrix.UB[1][2]
  (~ **_diffrn_orient_matrix_UB_12**)

  _diffrn_orient_matrix.UB[1][3]
  (~ **_diffrn_orient_matrix_UB_13**)

  _diffrn_orient_matrix.UB[2][1]
  (~ **_diffrn_orient_matrix_UB_21**)

  _diffrn_orient_matrix.UB[2][2]
  (~ **_diffrn_orient_matrix_UB_22**)

  _diffrn_orient_matrix.UB[2][3]
  (~ **_diffrn_orient_matrix_UB_23**)

  _diffrn_orient_matrix.UB[3][1]
  (~ **_diffrn_orient_matrix_UB_31**)

  _diffrn_orient_matrix.UB[3][2]
  (~ **_diffrn_orient_matrix_UB_32**)

  _diffrn_orient_matrix.UB[3][3]
  (~ **_diffrn_orient_matrix_UB_33**)

(*f*) DIFFRN_ORIENT_REFLN
- **_diffrn_orient_refln.diffrn_id**
  → **_diffrn.id**
- **_diffrn_orient_refln.index_h**
- **_diffrn_orient_refln.index_k**
- **_diffrn_orient_refln.index_l**

  _diffrn_orient_refln.angle_chi

  _diffrn_orient_refln.angle_kappa

  _diffrn_orient_refln.angle_omega

  _diffrn_orient_refln.angle_phi

  _diffrn_orient_refln.angle_psi

  _diffrn_orient_refln.angle_theta

(*g*) DIFFRN_RADIATION
- **_diffrn_radiation.diffrn_id**
  → **_diffrn.id**

  _diffrn_radiation.collimation

  _diffrn_radiation.filter_edge

  _diffrn_radiation.inhomogeneity

  _diffrn_radiation.monochromator

  _diffrn_radiation.polarisn_norm

  _diffrn_radiation.polarisn_ratio

  _diffrn_radiation.probe

  _diffrn_radiation.type

_diffrn_radiation.wavelength_id
→ _diffrn_radiation_wavelength.id
_diffrn_radiation.xray_symbol

(*h*) DIFFRN_RADIATION_WAVELENGTH
• _diffrn_radiation_wavelength.id
• _diffrn_radiation_wavelength.wavelength
(∼ _diffrn_radiation_wavelength.wavelength)
• _diffrn_radiation_wavelength.wt

(*i*) DIFFRN_REFLN
• _diffrn_refln.diffrn_id
→ _diffrn.id
• _diffrn_refln.id
_diffrn_refln.angle_chi
_diffrn_refln.angle_kappa
_diffrn_refln.angle_omega
_diffrn_refln.angle_phi
_diffrn_refln.angle_psi
_diffrn_refln.angle_theta
_diffrn_refln.attenuator_code
→ _diffrn_attenuator.code
_diffrn_refln.class_code
_diffrn_refln.counts_bg_1
_diffrn_refln.counts_bg_2
_diffrn_refln.counts_net
_diffrn_refln.counts_peak
_diffrn_refln.counts_total
_diffrn_refln.detect_slit_horiz
_diffrn_refln.detect_slit_vert
_diffrn_refln.elapsed_time
_diffrn_refln.index_h
_diffrn_refln.index_k
_diffrn_refln.index_l
_diffrn_refln.intensity_net
_diffrn_refln.intensity_sigma
_diffrn_refln.intensity_u
_diffrn_refln.scale_group_code
→ _diffrn_scale_group.code
_diffrn_refln.scan_mode
_diffrn_refln.scan_mode_backgd
_diffrn_refln.scan_rate
_diffrn_refln.scan_time_backgd
_diffrn_refln.scan_width
_diffrn_refln.sint_over_lambda
(∼ _diffrn_refln.sint/lambda)
_diffrn_refln.standard_code
→ _diffrn_standard_refln.code
_diffrn_refln.wavelength
_diffrn_refln.wavelength_id
→ _diffrn_radiation.wavelength_id

(*j*) DIFFRN_REFLNS
• _diffrn_reflns.diffrn_id
→ _diffrn.id
_diffrn_reflns.av_R_equivalents
_diffrn_reflns.av_sigmaI_over_netI
_diffrn_reflns.av_unetI/netI
_diffrn_reflns.limit_h_max
_diffrn_reflns.limit_h_min
_diffrn_reflns.limit_k_max
_diffrn_reflns.limit_k_min
_diffrn_reflns.limit_l_max
_diffrn_reflns.limit_l_min
_diffrn_reflns.number
_diffrn_reflns.reduction_process
_diffrn_reflns.theta_max
_diffrn_reflns.theta_min
_diffrn_reflns.transf_matrix[1][1]
(∼ _diffrn_reflns_transf_matrix_11)
_diffrn_reflns.transf_matrix[1][2]
(∼ _diffrn_reflns_transf_matrix_12)
_diffrn_reflns.transf_matrix[1][3]
(∼ _diffrn_reflns_transf_matrix_13)
_diffrn_reflns.transf_matrix[2][1]
(∼ _diffrn_reflns_transf_matrix_21)
_diffrn_reflns.transf_matrix[2][2]
(∼ _diffrn_reflns_transf_matrix_22)
_diffrn_reflns.transf_matrix[2][3]
(∼ _diffrn_reflns_transf_matrix_23)
_diffrn_reflns.transf_matrix[3][1]
(∼ _diffrn_reflns_transf_matrix_31)
_diffrn_reflns.transf_matrix[3][2]
(∼ _diffrn_reflns_transf_matrix_32)
_diffrn_reflns.transf_matrix[3][3]
(∼ _diffrn_reflns_transf_matrix_33)

(*k*) DIFFRN_REFLNS_CLASS
• _diffrn_reflns_class.code
_diffrn_reflns_class.av_R_eq
_diffrn_reflns_class.av_sgI/I
_diffrn_reflns_class.av_uI/I
_diffrn_reflns_class.d_res_high
_diffrn_reflns_class.d_res_low
_diffrn_reflns_class.description
_diffrn_reflns_class.number

(*l*) DIFFRN_SCALE_GROUP
• _diffrn_scale_group.code
_diffrn_scale_group.I_net

(*m*) DIFFRN_SOURCE
• _diffrn_source.diffrn_id
→ _diffrn.id
_diffrn_source.current
_diffrn_source.details
_diffrn_source.power
_diffrn_source.size
_diffrn_source.source (∼ _diffrn_source)
_diffrn_source.take-off_angle
_diffrn_source.target
_diffrn_source.type
_diffrn_source.voltage

(*n*) DIFFRN_STANDARD_REFLN
• _diffrn_standard_refln.code
• _diffrn_standard_refln.diffrn_id
→ _diffrn.id
_diffrn_standard_refln.index_h
_diffrn_standard_refln.index_k
_diffrn_standard_refln.index_l

(*o*) DIFFRN_STANDARDS
• _diffrn_standards.diffrn_id
→ _diffrn.id
_diffrn_standards.decay_%
_diffrn_standards.interval_count
_diffrn_standards.interval_time
_diffrn_standards.number
_diffrn_standards.scale_sigma
_diffrn_standards.scale_u

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ∼ symbol. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string _esd to the data name listed.

To a very great extent, data items in the DIFFRN category group are used in the same way in the mmCIF and core CIF dictionaries, and Section 3.2.2.2 can be consulted for details. Example 3.6.5.2 shows how these categories are used to describe the data collection for a macromolecule.

There is, however, one important difference. An mmCIF may describe several separate diffraction experiments that were conducted with a common purpose; each such experiment would be given a unique value of **_diffrn.id**, the key for the DIFFRN category. Descriptions of features of that experiment in related categories would be given a matching identifier with the same value (*e.g.* **_diffrn_detector.diffrn_id**). The use of the suffix ***.diffrn_id** for the key data names in each related category emphasizes the connection to the parent experiment.

As a consequence, there are differences between the mmCIF and core CIF dictionaries in the definition of the category keys for

Example 3.6.5.2. *Data collection for an HIV-1 protease crystal (PDB 5HVP) described with data items in the* DIFFRN *and related categories.*

```
_diffrn.id                              'set1'
_diffrn.crystal_id                       1
_diffrn.ambient_temp                     293(3)
_diffrn.ambient_environment
; Mother liquor from the reservoir of the vapor
  diffusion experiment, mounted in room air
;
_diffrn.crystal_support
;  0.7 mm glass capillary, sealed with dental wax
;
_diffrn.crystal_treatment
; Equilibrated in rotating anode radiation enclosure
  for 18 hours prior to beginning of data collection.
;
_diffrn_detector.diffrn_id               'set1'
_diffrn_detector.detector                'multiwire'
_diffrn_detector.type                    'Siemens'

_diffrn_measurement.diffrn_id            'd1'
_diffrn_measurement.device          '3-circle camera'
_diffrn_measurement.device_type      'Supper model x'
_diffrn_measurement.device_details       'none'
_diffrn_measurement.method               'omega scan'
_diffrn_measurement.details
; 440 frames, 0.20 degrees, 150 sec, detector
  distance 12 cm, detector angle 22.5 degrees
;
_diffrn_radiation.diffrn_id               'set1'
_diffrn_radiation.collimation
                              '0.3 mm double pinhole'
_diffrn_radiation.monochromator           'graphite'
_diffrn_radiation.type                    'Cu Kalpha'
_diffrn_radiation.wavelength_id            1
_diffrn_radiation_wavelength.id            1
_diffrn_radiation_wavelength.wavelength 1.54
_diffrn_radiation_wavelength.wt            1.0
_diffrn_source.diffrn_id               'set1'
_diffrn_source.source                  'rotating anode'
_diffrn_source.type                    'Rigaku RU-200'
_diffrn_source.power                    50
_diffrn_source.current                  180
_diffrn_source.target       '8mm x 0.4 mm broad-focus'
```

the DIFFRN categories. These differences were introduced in order to accommodate data from more than one experiment in the same table. For example, in the core CIF dictionary, the Miller indices `_diffrn_refln_index_h`, `*_k` and `*_l` play the role of the category key for the DIFFRN_REFLN category. In the mmCIF dictionary, the category key is formed by the data items `_diffrn_refln.id` and `_diffrn_refln.diffrn_id`.

3.6.5.3. Growth, description and analysis of the crystal

The categories describing the crystal properties and growth are as follows:

EXPTL group
- *Crystal properties* (§3.6.5.3.1)
  - EXPTL
  - EXPTL_CRYSTAL
  - EXPTL_CRYSTAL_FACE
- *Crystal growth* (§3.6.5.3.2)
  - EXPTL_CRYSTAL_GROW
  - EXPTL_CRYSTAL_GROW_COMP

Categories in the EXPTL category group are used to describe experimental measurements on the crystal (*e.g.* of its shape, size and density) and the growth of the crystal. Data items in the EXPTL category are used to describe the gross properties of the crystal or crystals used in the experiment. Data items in the EXPTL_CRYSTAL category are used to describe the crystal properties in detail and allow for cases where multiple crystals are used. The data items in the EXPTL_CRYSTAL_FACE category are used to describe the crystal faces.

Data items for describing crystal growth are given in two categories that are not found in the current version of the core CIF dictionary. Data items in the EXPTL_CRYSTAL_GROW category are used to describe the conditions and methods used to grow the crystals, and data items in the EXPTL_CRYSTAL_GROW_COMP category can be used to list the components of the solutions in which the crystals were grown.

3.6.5.3.1. *Crystal properties*

The data items in these categories are as follows:

(*a*) EXPTL

- `_exptl.entry_id`
  → `_entry.id`
  `_exptl.absorpt_coefficient_mu`
  `_exptl.absorpt_correction_T_max`
  `_exptl.absorpt_correction_T_min`
  `_exptl.absorpt_correction_type`
  `_exptl.absorpt_process_details`
  `_exptl.crystals_number`
  `_exptl.details` (∼ `_exptl_special_details`)
  `_exptl.method`
  `_exptl.method_details`

(*b*) EXPTL_CRYSTAL

- `_exptl_crystal.id`
  `_exptl_crystal.colour`
  `_exptl_crystal.colour_lustre`
  `_exptl_crystal.colour_modifier`
  `_exptl_crystal.colour_primary`
  `_exptl_crystal.density_diffrn`
  `_exptl_crystal.density_Matthews`
+ `_exptl_crystal.density_meas`
  `_exptl_crystal.density_meas_gt`
  `_exptl_crystal.density_meas_lt`
+ `_exptl_crystal.density_meas_temp`
  `_exptl_crystal.density_meas_temp_gt`
  `_exptl_crystal.density_meas_temp_lt`
  `_exptl_crystal.density_method`
  `_exptl_crystal.density_percent_sol`
  `_exptl_crystal.description`
  `_exptl_crystal.F_000`
  `_exptl_crystal.preparation`
  `_exptl_crystal.size_max`
  `_exptl_crystal.size_mid`
  `_exptl_crystal.size_min`
  `_exptl_crystal.size_rad`

(*c*) EXPTL_CRYSTAL_FACE

- `_exptl_crystal_face.crystal_id`
  → `_exptl_crystal.id`
- `_exptl_crystal_face.index_h`
- `_exptl_crystal_face.index_k`
- `_exptl_crystal_face.index_l`
  `_exptl_crystal_face.diffr_chi`
  `_exptl_crystal_face.diffr_kappa`
  `_exptl_crystal_face.diffr_phi`
  `_exptl_crystal_face.diffr_psi`
  `_exptl_crystal_face.perp_dist`

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (`.`) *to an underscore* (_) *except where indicated by the* ∼ *symbol. Data items marked with a plus* (+) *have companion data names for the standard uncertainty in the reported value, formed by appending the string* `_esd` *to the data name listed.*

Data items in these categories are used in the same way in the mmCIF and core CIF dictionaries, and Section 3.2.2.3 can be consulted for details (see Example 3.6.5.3). Identifiers have been introduced to the categories to provide the formal category keys required by the DDL2 data model.

Example 3.6.5.3. *The crystal used in the determination of an HIV-1 protease structure (PDB 5HVP) described using data items in the* EXPTL *and* EXPTL_CRYSTAL *categories.*

```
_exptl.entry_id                    '5HVP'
_exptl.crystals_number             1
_exptl.method     'single-crystal x-ray diffraction'
_exptl.method_details
; graphite monochromatized Cu K(alpha) fixed tube
  and Siemens multiwire detector used
;
_exptl_crystal.id                  1
_exptl_crystal.colour              'colorless'
_exptl_crystal.density_percent_sol0.57
_exptl_crystal.description        'rectangular plate'
_exptl_crystal.size_max            0.30
_exptl_crystal.size_mid            0.20
_exptl_crystal.size_min            0.05
```

3.6.5.3.2. *Crystal growth*

The data items in these categories are as follows:

(*a*) EXPTL_CRYSTAL_GROW

```
• _exptl_crystal_grow.crystal_id
        → _exptl_crystal.id
  _exptl_crystal_grow.apparatus
  _exptl_crystal_grow.atmosphere
  _exptl_crystal_grow.details
  _exptl_crystal_grow.method
  _exptl_crystal_grow.method_ref
  _exptl_crystal_grow.pH
+ _exptl_crystal_grow.pressure
  _exptl_crystal_grow.seeding
  _exptl_crystal_grow.seeding_ref
+ _exptl_crystal_grow.temp
  _exptl_crystal_grow.temp_details
  _exptl_crystal_grow.time
```

(*b*) EXPTL_CRYSTAL_GROW_COMP

```
• _exptl_crystal_grow_comp.crystal_id
        → _exptl_crystal.id
• _exptl_crystal_grow_comp.id
  _exptl_crystal_grow_comp.conc
  _exptl_crystal_grow_comp.details
  _exptl_crystal_grow_comp.name
  _exptl_crystal_grow_comp.sol_id
  _exptl_crystal_grow_comp.volume
```

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Data items marked with a plus* (+) *have companion data names for the standard uncertainty in the reported value, formed by appending the string* `_esd` *to the data name listed.*

Crystallization strategies and protocols are very varied and may not lend themselves to a formal tabulation. Common or well defined techniques may be indicated using the data item `_exptl_crystal_grow.method`, and a literature reference, where appropriate, may be given using `_exptl_crystal_grow.method_ref`. Frequently, however, a detailed description of methodology is required; this can be given in `_exptl_crystal_grow.details`. Example 3.6.5.4 shows how information about strategies that were attempted and proved unsuccessful can be recorded. In circumstances such as this, the data item `_exptl_crystal_grow.pH` would record the final pH.

Where the crystallization protocol is well defined, it is useful to list the individual components of the solution in the category EXPTL_CRYSTAL_GROW_COMP. Example 3.6.5.4 labels the solutions used as 1 and 2, in accordance with the convention that solution 1 contains the molecule to be crystallized and solution 2 (and if necessary additional solutions) contains the precipitant. However, it is permissible and may be preferable to use more explicit labels such as 'well solution' in the `_exptl_crystal_grow_comp.sol_id` field.

Example 3.6.5.4. *The growth of HIV-1 protease crystals (PDB 5HVP) described with data items in the* EXPTL_CRYST_GROW *and* EXPTL_CRYSTAL_GROW_COMP *categories.*

```
_exptl_crystal_grow.crystal_id     1
_exptl_crystal_grow.method         'hanging drop'
_exptl_crystal_grow.apparatus      'Linbro plates'
_exptl_crystal_grow.atmosphere     'room air'
_exptl_crystal_grow.pH             4.7
_exptl_crystal_grow.temp           18(3)
_exptl_crystal_grow.time    'approximately 2 days'
_exptl_crystal_grow.details
; The dependence on pH for successful crystal growth
  is very sharp. At pH 7.4 only showers of tiny
  crystals grew, at pH 7.5 well formed single
  crystals grew, at pH 7.6 no crystallization
  occurred at all.
;
loop_
  _exptl_crystal_grow_comp.crystal_id
  _exptl_crystal_grow_comp.id
  _exptl_crystal_grow_comp.sol_id
  _exptl_crystal_grow_comp.name
  _exptl_crystal_grow_comp.volume
  _exptl_crystal_grow_comp.conc
  _exptl_crystal_grow_comp.details
  1  1  1  'HIV-1 protease'  '0.002 ml'  '6 mg/ml'
; The protein solution was in a buffer containing
  25 mM NaCl, 100 mM NaMES/MES buffer, pH 7.5,
  3 mM NaAzide
;
  1  2  2  'NaCl'  '0.200 ml'  '4 M'
  'in 3 mM NaAzide'
  1  3  2  'Acetic Acid'  '0.047 ml'  '100 mM'
  'in 3 mM NaAzide'
  1  4  2  'Na Acetate'  '0.053 ml'  '100 mM'
; in 3 mM NaAzide. Buffer components were mixed
  to produce a pH of 4.7 according to a ratio
  calculated from the pKa. The actual pH of
  solution 2 was not measured.
;
  1  5  2  'water'  '0.700 ml'  'neat'
 'in 3 mM NaAzide'
```

3.6.6. Analysis

The mmCIF dictionary contributes several new categories and data items to the REFINE and REFLN category groups. These reflect common practices in macromolecular crystallography in refinement and in the handling of experimental observations.

A new category group, the PHASING group, has been introduced to provide a structured description of phasing strategies, as macromolecular crystallography differs strongly from small-molecule crystallography in how phases are determined. The data model for phasing in the current version of the mmCIF dictionary cannot describe all approaches to phasing yet. Additions and revisions to the data items in the PHASING group of categories are anticipated in future versions of the dictionary.

3.6.6.1. Phasing

The categories describing phasing are as follows:

PHASING group

Overall description of phasing (§3.6.6.1.1)

PHASING

Phasing via molecular averaging (§3.6.6.1.2)

PHASING_AVERAGING

Phasing via isomorphous replacement (§3.6.6.1.3)

PHASING_ISOMORPHOUS

Phasing via multiple-wavelength anomalous dispersion (§3.6.6.1.4)

PHASING_MAD

PHASING_MAD_CLUST

PHASING_MAD_EXPT
PHASING_MAD_RATIO
PHASING_MAD_SET

Phasing via multiple isomorphous replacement (§3.6.6.1.5)

PHASING_MIR
PHASING_MIR_DER
PHASING_MIR_DER_REFLN
PHASING_MIR_DER_SHELL
PHASING_MIR_DER_SITE
PHASING_MIR_DER_SHELL

Phasing data sets (§3.6.6.1.6)

PHASING_SET
PHASING_SET_REFLN

The data items in the PHASING category group can be used to record details about the phasing of the structure and cover the various methods used in the phasing process. Many data items are provided for multiple isomorphous replacement (MIR) and multiple-wavelength anomalous dispersion (MAD). More limited sets of data items are provided for phasing using molecular averaging and phasing *via* using a structure that is isomorphous to the present structure. The current version of the mmCIF dictionary does not provide specific data items for recording the details of phasing *via* molecular replacement.

3.6.6.1.1. *Overall description of phasing*

The single data item in this category is as follows:

PHASING
• **_phasing.method**

The bullet (•) *indicates a category key.*

Phasing of macromolecular structures often involves the application of more than one of the methods described in the PHASING section of the mmCIF dictionary, such as when phases generated from a multiple isomorphous replacement experiment are improved by molecular averaging. The PHASING category is used to list the methods that were used.

At present, the category contains a single data item, the purpose of which is to specify the method employed in the structure determination. It may have one or more of the values listed in the dictionary (Example 3.6.6.1).

3.6.6.1.2. *Phasing via molecular averaging*

The data items in this category are as follows:

PHASING_AVERAGING
• **_phasing_averaging.entry_id**
  → _entry.id
  _phasing_averaging.details
  _phasing_averaging.method

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

When more than one copy of a molecule is present in the asymmetric unit, phases can be improved by averaging an electron-density map over the multiple images of the molecule. In some special cases with very high noncrystallographic symmetry, *de novo* phases have been derived by iterative application of molecular averaging, but more often averaging is used to improve phases determined by another method.

There are many protocols used for phasing with averaging and they are very varied. It was not thought to be appropriate to specify data items for any one approach in the current version of the mmCIF dictionary. The data items that are provided allow a text-based description of the protocol to be given; a formalism for recording a fully parsable description of molecular averaging needs to be developed for future revisions of the dictionary.

Example 3.6.6.1. *The methods used to generate the phases for a hypothetical structure described with the data item in the* PHASING *category.*

```
loop_
_phasing.method
   'mir'
   'averaging'
```

Example 3.6.6.2. *Phase improvement with molecular averaging for a hypothetical structure described with data items in the* PHASING_AVERAGING *category.*

```
_phasing_averaging.entry_id     'EXAMHYPO'
_phasing_averaging.method
; Iterative threefold averaging alternating with
  phase extensions by 0.5 reciprocal lattice units
  per cycle.
;

_phasing_averaging.details
; The position of the threefold axis was redetermined
  every five cycles.
;
```

Data items in the PHASING_AVERAGING category allow free-text descriptions to be given of the method used for structure determination or phase improvement using averaging over multiple observations of the molecule in the asymmetric unit and of any specific details of the application of the method to the current structure determination (Example 3.6.6.2). Note that the reference to the method is to be used to describe the method itself, and not as a reference to a software package; references to software packages would be made using data items in the SOFTWARE category.

3.6.6.1.3. *Phasing via isomorphous replacement*

The data items in this category are as follows:

PHASING_ISOMORPHOUS
• **_phasing_isomorphous.entry_id**
  → _entry.id
  _phasing_isomorphous.details
  _phasing_isomorphous.method
  _phasing_isomorphous.parent

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

Phases for many macromolecular structures are obtained from a previous determination of the same structure in the same crystal lattice. Examples of this are the determination of the structure of a point mutant or the determination of a structure in which a ligand is bound to an active site that was empty in the previous structure determination. In these cases, the new structure is essentially isomorphous with the parent structure, hence this method of phasing is termed 'isomorphous phasing' in the mmCIF dictionary. It is not to be confused with multiple isomorphous phasing (MIR), a phasing technique that involves the use of heavy-atom derivatives. MIR phasing is discussed in Section 3.6.6.1.5.

Not much information is needed to characterize isomorphous phasing. The 'parent' structure (the structure used to generate the initial phases for the present structure) is described in a free-text field and a second free-text field can be used to give details of the application of the method to the determination of the present structure (for instance, the removal of solvent or a bound ligand). In Example 3.6.6.3, the parent structure is the PDB entry 5HVP and the structure that is the subject of the present data block is identified as 'HVP+CmpdA'. **_phasing_isomorphous.method** allows

Example 3.6.6.3. *Isomorphous replacement phasing of an HIV-1 protease structure described using data items in the* PHASING_ISOMORPHOUS *category.*

```
_phasing_isomorphous.entry_id     'HVP+CmpdA'
_phasing_isomorphous.parent       'PDB entry 5HVP'
_phasing_isomorphous.details
; The inhibitor and all solvent atoms were removed
  from the parent structure before beginning
  refinement. All static disorder present in the
  parent structure was also removed.
;
```

any formal techniques that were used in the application of the method to the present structure determination to be described, for example rigid-body refinement. Note that this data item is not to be used to reference a software package; this would be done using data items in the SOFTWARE category.

3.6.6.1.4. *Phasing via multiple-wavelength anomalous dispersion*

The data items in these categories are as follows:

(*a*) PHASING_MAD

- _phasing_MAD.entry_id
  → _entry.id
  _phasing_MAD.details
  _phasing_MAD.method

(*b*) PHASING_MAD_CLUST

- _phasing_MAD_clust.expt_id
  → _phasing_MAD_clust.expt_id
- _phasing_MAD_clust.id
  _phasing_MAD_clust.number_set

(*c*) PHASING_MAD_EXPT

- _phasing_MAD_expt.id
  _phasing_MAD_expt.delta_delta_phi
  _phasing_MAD_expt.delta_phi
  _phasing_MAD_expt.delta_phi_sigma
  _phasing_MAD_expt.mean_fom
  _phasing_MAD_expt.number_clust
  _phasing_MAD_expt.R_normal_all
  _phasing_MAD_expt.R_normal_anom_scat

(*d*) PHASING_MAD_RATIO

- _phasing_MAD_ratio.expt_id
  → _phasing_MAD_expt.id
- _phasing_MAD_ratio.clust_id
  → _phasing_MAD_clust.id
- _phasing_MAD_ratio.wavelength_1
  → _phasing_MAD_set.wavelength
- _phasing_MAD_ratio.wavelength_2
  → _phasing_MAD_set.wavelength
  _phasing_MAD_ratio.d_res_high
  _phasing_MAD_ratio.d_res_low
  _phasing_MAD_ratio.ratio_one_wl
  _phasing_MAD_ratio.ratio_one_wl_centric
  _phasing_MAD_ratio.ratio_two_wl

(*e*) PHASING_MAD_SET

- _phasing_MAD_set.clust_id
  → _phasing_MAD_clust.id
- _phasing_MAD_set.expt_id
  → _phasing_MAD_expt.id
- _phasing_MAD_set.set_id
  → _phasing_set.id
- _phasing_MAD_set.wavelength
  _phasing_MAD_set.d_res_high
  _phasing_MAD_set.d_res_low
  _phasing_MAD_set.f_double_prime
  _phasing_MAD_set.f_prime
  _phasing_MAD_set.wavelength_details

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

PHASING_MAD and related categories are used to provide information about phasing using the multiple-wavelength anomalous dispersion (MAD) technique. The data model used for MAD phasing in the current version of the mmCIF dictionary is that of Hendrickson, as exemplified in the structure determination of N-cadherin (Shapiro *et al.*, 1995; Example 3.6.6.4). In current practice, MAD phasing is often treated as a special case of MIR phasing and the PHASING_MIR categories would be more appropriate to describe the results.

Unlike the PHASING_MIR categories, there is no provision in the current mmCIF model of MAD phasing for analysis of the overall phasing statistics and the contribution to the phasing of each data set by bins of resolution, and no provision for giving a list of the phased reflections. This will need to be addressed in future versions of the mmCIF dictionary.

Example 3.6.6.4. *MAD phasing of the structure of N-cadherin (Shapiro et al., 1995) described using data items in the* PHASING_MAD *and related categories.*

```
_phasing_MAD.entry_id       'NCAD'

loop_
_phasing_MAD_expt.id
_phasing_MAD_expt.number_clust
_phasing_MAD_expt.R_normal_all
_phasing_MAD_expt.R_normal_anom_scat
_phasing_MAD_expt.delta_delta_phi
_phasing_MAD_expt.delta_phi_sigma
_phasing_MAD_expt.mean_fom
   1  2  0.063  0.451  58.5  20.3  0.88
   2  1  0.051  0.419  36.8  18.2  0.93

loop_
_phasing_MAD_clust.id
_phasing_MAD_clust.expt_id
_phasing_MAD_clust.number_set
  'four wavelength'  1  4
  'five wavelength'  1  5
  'five wavelength'  2  5

loop_
_phasing_MAD_ratio.expt_id
_phasing_MAD_ratio.clust_id
_phasing_MAD_ratio.wavelength_1
_phasing_MAD_ratio.wavelength_2
_phasing_MAD_ratio.d_res_low
_phasing_MAD_ratio.d_res_high
_phasing_MAD_ratio.ratio_two_wl
_phasing_MAD_ratio.ratio_one_wl
_phasing_MAD_ratio.ratio_one_wl_centric
   1  'four wavelength'  1.4013 1.4013  20.00  4.00
        .  0.084 0.076
   1  'four wavelength'  1.4013 1.3857  20.00  4.00
     0.067      .     .
   1  'four wavelength'  1.4013 1.3852  20.00  4.00
     0.051      .     .
   1  'four wavelength'  1.4013 1.3847  20.00  4.00
     0.044      .     .
   1  'four wavelength'  1.3857 1.3857  20.00  4.00
        .  0.110 0.049
   1  'four wavelength'  1.3857 1.3852  20.00  4.00
     0.049      .     .
# - - - abbreviated - - -

loop_
_phasing_MAD_set.expt_id
_phasing_MAD_set.clust_id
_phasing_MAD_set.set_id
_phasing_MAD_set.wavelength
_phasing_MAD_set.wavelength_details
_phasing_MAD_set.d_res_low
_phasing_MAD_set.d_res_high
_phasing_MAD_set.f_prime
_phasing_MAD_set.f_double_prime
   1 'four wavelength' aa 1.4013 'pre-edge'  20.00
      3.00   -12.48   3.80
   1 'four wavelength' bb 1.3857 'peak'      20.00
      3.00.  -31.22  17.20
   1 'four wavelength' cc 1.3852 'edge'      20.00
      3.00   -13.97  29.17
```

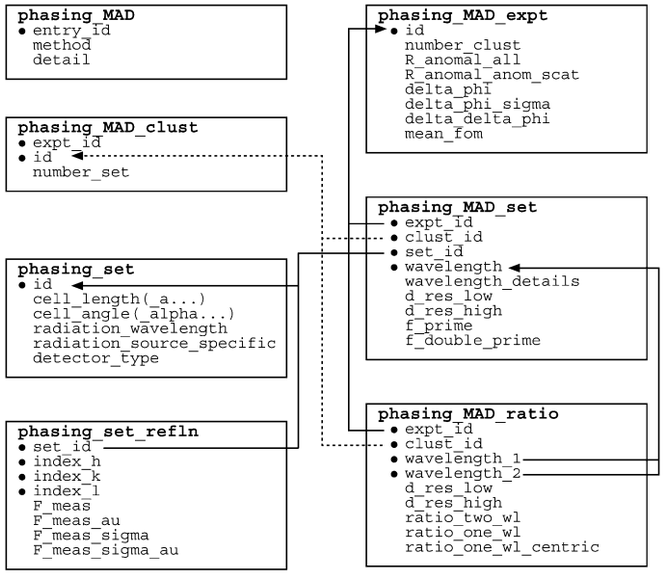


Fig. 3.6.6.1. The family of categories used to describe MAD phasing. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

The relationships between categories describing MAD phasing are shown in Fig. 3.6.6.1.

Data items in the PHASING_MAD category allow a brief overview of the method that was used to be given and allow special aspects of the phasing strategy to be noted; data items in this category are analogous to the data items in the other overview categories describing phasing techniques.

In the data model for MAD phasing used in the present version of the mmCIF dictionary, a collection of data sets measured at different wavelengths can be used to construct more than one set of phases. These phase sets will produce electron-density maps with different local properties. The model of the structure is often constructed using information from a collection of these maps. The collections of multiple phase sets are referred to as 'experiments' and the groups of data sets that contribute to each experiment are referred to as 'clusters'. Data items in PHASING_MAD_EXPT identify each experiment and give the number of contributing clusters. Additional data items record the phase difference between the structure factors due to normal scattering from all atoms and from only the anomalous scatterers, the standard uncertainty of this quantity, the mean figure of merit, and a number of other indicators of the quality of the phasing.

Data items in the PHASING_MAD_CLUST category can be used to label the clusters of data sets and give the number of data sets allocated to each cluster. In Example 3.6.6.4 two experiments are described. The first experiment contains two clusters, one of which contains four data sets and the second of which contains five data sets. The second experiment contains a single cluster of five data sets. Note that the author has chosen informative labels to identify the clusters ('four wavelength', 'five wavelength'). Carefully chosen labels can help someone reading the mmCIF to trace the complex relationships between the categories.

Data items in the PHASING_MAD_RATIO category can be used to record the ratios of phasing statistics (Bijvoet differences) between pairs of data sets in a MAD phasing experiment, within shells of resolution characterized by **_phasing_MAD_ratio.d_res_high** and ***.d_res_low**.

The data sets used in the MAD phasing experiments are described using data items in the PHASING_MAD_SET category. Each data set is characterized by resolution shell and wavelength, and by the f' and f'' components of the anomalous scattering factor at that wavelength. The actual observations in each data set and the experimental conditions under which they were made are recorded using data items in the PHASING_SET and PHASING_SET_REFLN categories.

3.6.6.1.5. *Phasing via multiple isomorphous replacement*

The data items in these categories are as follows:

(*a*) PHASING_MIR

- **_phasing_MIR.entry_id**
  → **_entry.id**

_phasing_MIR.details
_phasing_MIR.d_res_high
_phasing_MIR.d_res_low
_phasing_MIR.FOM
_phasing_MIR.FOM_acentric
_phasing_MIR.FOM_centric
_phasing_MIR.method
_phasing_MIR.reflns
_phasing_MIR.reflns_acentric
_phasing_MIR.reflns_centric
_phasing_MIR.reflns_criterion

(*b*) PHASING_MIR_SHELL

- **_phasing_MIR_shell.d_res_high**
- **_phasing_MIR_shell.d_res_low**

_phasing_MIR_shell.FOM
_phasing_MIR_shell.FOM_acentric
_phasing_MIR_shell.FOM_centric
_phasing_MIR_shell.loc
_phasing_MIR_shell.mean_phase
_phasing_MIR_shell.power
_phasing_MIR_shell.R_cullis
_phasing_MIR_shell.R_kraut
_phasing_MIR_shell.reflns
_phasing_MIR_shell.reflns_acentric
_phasing_MIR_shell.reflns_anomalous
_phasing_MIR_shell.reflns_centric

(*c*) PHASING_MIR_DER

- **_phasing_MIR_der.id**

_phasing_MIR_der.d_res_high
_phasing_MIR_der.d_res_low
_phasing_MIR_der.der_set_id
→ **_phasing_set.id**
_phasing_MIR_der.details
_phasing_MIR_der.native_set_id
→ **_phasing_set.id**
_phasing_MIR_der.number_of_sites
_phasing_MIR_der.power_acentric
_phasing_MIR_der.power_centric
_phasing_MIR_der.R_cullis_acentric
_phasing_MIR_der.R_cullis_anomalous
_phasing_MIR_der.R_cullis_centric
_phasing_MIR_der.reflns_acentric
_phasing_MIR_der.reflns_anomalous
_phasing_MIR_der.reflns_centric
_phasing_MIR_der.reflns_criteria

(*d*) PHASING_MIR_DER_REFLN

- **_phasing_MIR_der_refln.der_id**
  → **_phasing_MIR_der.id**
- **_phasing_MIR_der_refln.index_h**
- **_phasing_MIR_der_refln.index_k**
- **_phasing_MIR_der_refln.index_l**
- **_phasing_MIR_der_refln.set_id**
  → **_phasing_set.id**

_phasing_MIR_der_refln.F_calc
_phasing_MIR_der_refln.F_calc_au
_phasing_MIR_der_refln.F_meas
_phasing_MIR_der_refln.F_meas_au
_phasing_MIR_der_refln.F_meas_sigma
_phasing_MIR_der_refln.F_meas_sigma_au
_phasing_MIR_der_refln.HL_A_iso
_phasing_MIR_der_refln.HL_B_iso
_phasing_MIR_der_refln.HL_C_iso

```
  _phasing_MIR_der_refln.HL_D_iso
  _phasing_MIR_der_refln.phase_calc

(e) PHASING_MIR_DER_SHELL
• _phasing_MIR_der_shell.d_res_high
• _phasing_MIR_der_shell.d_res_low
• _phasing_MIR_der_shell.der_id
        → _phasing_MIR_der.id
  _phasing_MIR_der_shell.fom
  _phasing_MIR_der_shell.ha_ampl
  _phasing_MIR_der_shell.loc
  _phasing_MIR_der_shell.phase
  _phasing_MIR_der_shell.power
  _phasing_MIR_der_shell.R_cullis
  _phasing_MIR_der_shell.R_kraut
  _phasing_MIR_der_shell.reflns

(f) PHASING_MIR_DER_SITE
• _phasing_MIR_der_site.der_id
        → _phasing_MIR_der.id
• _phasing_MIR_der_site.id
  _phasing_MIR_der_site.atom_type_symbol
        → _atom_type.symbol
+ _phasing_MIR_der_site.B_iso
+ _phasing_MIR_der_site.Cartn_x
+ _phasing_MIR_der_site.Cartn_y
+ _phasing_MIR_der_site.Cartn_z
  _phasing_MIR_der_site.details
+ _phasing_MIR_der_site.fract_x
+ _phasing_MIR_der_site.fract_y
+ _phasing_MIR_der_site.fract_z
  _phasing_MIR_der_site.occupancy
  _phasing_MIR_der_site.occupancy_anom
  _phasing_MIR_der_site.occupancy_anom_su
  _phasing_MIR_der_site.occupancy_iso
  _phasing_MIR_der_site.occupancy_iso_su
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string **_esd** *to the data name listed.*

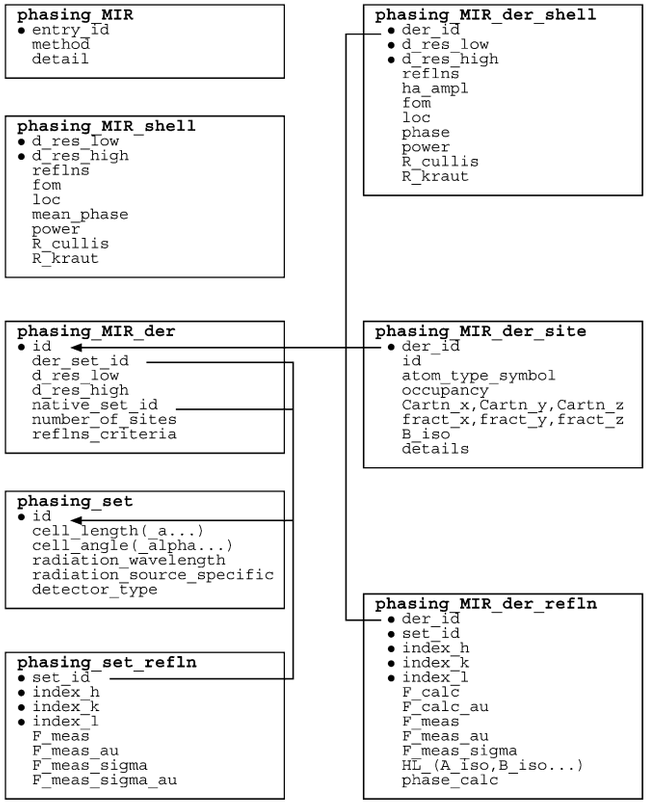


Fig. 3.6.6.2. The family of categories used to describe MIR phasing. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

PHASING_MIR and related categories provide information about phasing by methods involving multiple isomorphous replacement (MIR). These same categories may also be used to describe phasing by related techniques, such as single isomorphous replacement (SIR) and single or multiple isomorphous replacement plus anomalous scattering (SIRAS, MIRAS). The relationships between the categories describing MIR phasing are shown in Fig. 3.6.6.2.

As with the other overview categories described in this section, the PHASING_MIR category contains data items that can be used for text-based descriptions of the method used and any special aspects of its application. There are also items for describing the resolution limit of the reflections that were phased, the figures of merit for all reflections and for the acentric reflections phased in the native data set, and the total numbers of reflections and their inclusion threshold in the native data set. Statistics for the phasing can be given by shells of resolution using data items in the PHASING_MIR_SHELL category.

An MIR phasing experiment involves one or more derivatives. The remaining categories in this group are used to describe aspects of each derivative (Example 3.6.6.5). A derivative in this context does not necessarily correspond to a data set; for instance, the same data set could be used to one resolution limit as an isomorphous scatterer and to a different resolution (and with a different sigma cutoff) as an anomalous scatterer. These would be treated as two distinct derivatives, although both derivatives would point to the same data sets *via* **_phasing_MIR_der.der_set_id** and **_phasing_MIR_der.native_set_id** (see Fig. 3.6.6.2).

Data items in the PHASING_MIR_DER category can be used to identify and describe each derivative. The resolution limits for the individual derivatives need not match those of the overall phasing experiment, as the phasing power of each derivative as a function of resolution will vary. Many of the statistical descriptors of phasing given in the PHASING_MIR category are repeated in this category, as derivatives vary in quality and their contribution to the phasing must be assessed individually. These same statistical measures can be given for shells of resolution in the PHASING_MIR_DER_SHELL category.

Data items in the PHASING_MIR_DER_REFLN category can be used to provide details of each reflection used in an MIR phasing experiment. The pointer **_phasing_MIR_der_refln.set_id** links the reflection to a particular set of experimental data and **_phasing_MIR_der_refln.der_id** points to a particular derivative used in the phasing (as mentioned above, derivatives in this context do not equate to data sets). The phase assigned to each reflection and the measured and calculated values of its structure factor can be given. (It is not necessary to include the measured values of the structure factors in this list, since they are accessible in the PHASING_SET_REFLN category, but it may be convenient to present them here). Data items are also provided for the *A*, *B*, *C* and *D* phasing coefficients of Hendrickson & Lattman (1970).

The heavy atoms identified in each derivative can be listed using data items in the PHASING_MIR_DER_SITE category. Most of the data names are clear analogues of similar items in the ATOM_SITE category; an exception is **_phasing_MIR_der_site.occupancy_anom**, which specifies the relative anomalous occupancy of the atom type present at a heavy-atom site in a particular derivative.

Example 3.6.6.5. *Phasing of the structure of bovine plasma retinol-binding protein (Zanotti et al., 1993) described using data items in the* PHASING_MIR *and related categories.*

```
_phasing_MIR.entry_id          '1HBP'
_phasing_MIR.method
;  Standard phase refinement (Blow & Crick, 1959)
;

loop_
_phasing_MIR_shell.d_res_low
_phasing_MIR_shell.d_res_high
_phasing_MIR_shell.reflns
_phasing_MIR_shell.FOM
 15.0  8.3   80  0.69       8.3  6.4  184  0.73
  6.4  5.2  288  0.72       5.2  4.4  406  0.65
  4.4  3.8  554  0.54       3.8  3.4  730  0.53
  3.4  3.0  939  0.50

loop_
_phasing_MIR_der.id
_phasing_MIR_der.number_of_sites
_phasing_MIR_der.details
KAu(CN)2  3
     'major site interpreted in difference Patterson'
K2HgI4    6 'sites found in cross-difference Fourier'
K3IrCl6   2 'sites found in cross-difference Fourier'
All      11 'data for all three derivatives combined'

loop_
_phasing_MIR_der_shell.der_id
_phasing_MIR_der_shell.d_res_low
_phasing_MIR_der_shell.d_res_high
_phasing_MIR_der_shell.ha_ampl
_phasing_MIR_der_shell.loc
   KAu(CN)2  15.0  8.3   54   26
   KAu(CN)2   8.3  6.4   54   20
# - - - abbreviated - - -
   K2HgI4    15.0  8.3  149   87
   K2HgI4     8.3  6.4  121   73
# - - - abbreviated - - -
   K3IrCl6   15.0  8.3   33   27
   K3IrCl6    8.3  6.4   40   23
# - - - abbreviated - - -

loop_
_phasing_MIR_der_site.der_id
_phasing_MIR_der_site.id
_phasing_MIR_der_site.atom_type_symbol
_phasing_MIR_der_site.occupancy
_phasing_MIR_der_site.fract_x
_phasing_MIR_der_site.fract_y
_phasing_MIR_der_site.fract_z
_phasing_MIR_der_site.B_iso
   KAu(CN)2  1  Au  0.40  0.082  0.266  0.615  33.0
   KAu(CN)2  2  Au  0.03  0.607  0.217  0.816  25.9
   K2HgI4    1  Hg  0.63  0.048  0.286  0.636  33.7
   K2HgI4    2  Hg  0.34  0.913  0.768  0.889  36.7
# - - - abbreviated - - -

_phasing_MIR_der_refln.index_h           6
_phasing_MIR_der_refln.index_k           1
_phasing_MIR_der_refln.index_l          25
_phasing_MIR_der_refln.der_id        HGPT1
_phasing_MIR_der_refln.set_id      'NS1-96'
_phasing_MIR_der_refln.F_calc_au    106.66
_phasing_MIR_der_refln.F_meas_au    204.67
_phasing_MIR_der_refln.F_meas_sigma   6.21
_phasing_MIR_der_refln.HL_A_iso      -3.15
_phasing_MIR_der_refln.HL_B_iso      -0.76
_phasing_MIR_der_refln.HL_C_iso       0.65
_phasing_MIR_der_refln.HL_D_iso       0.23
_phasing_MIR_der_refln.phase_calc   194.48
```

3.6.6.1.6. *Phasing data sets*

The data items in these categories are as follows:

(*a*) PHASING_SET

```
• _phasing_set.id
  _phasing_set.cell_angle_alpha
  _phasing_set.cell_angle_beta
  _phasing_set.cell_angle_gamma
  _phasing_set.cell_length_a
  _phasing_set.cell_length_b
  _phasing_set.cell_length_c
  _phasing_set.detector_specific
  _phasing_set.detector_type
  _phasing_set.radiation_source_specific
  _phasing_set.radiation_wavelength
  _phasing_set.temp
```

(*b*) PHASING_SET_REFLN

```
• _phasing_set_refln.index_h
• _phasing_set_refln.index_k
• _phasing_set_refln.index_l
• _phasing_set_refln.set_id
        → _phasing_set.id
  _phasing_set_refln.F_meas
  _phasing_set_refln.F_meas_au
  _phasing_set_refln.F_meas_sigma
  _phasing_set_refln.F_meas_sigma_au
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

Data items in the PHASING_SET family of categories are homologous to items with related names in the CELL and DIFFRN families of categories. The PHASING_SET categories were added to the mmCIF data model so that intensity and phase information for the data sets used in phasing could be stored in the same data block as the information for the refined structure. It is not necessary to store all the experimental information for each data set (*e.g.* the raw data sets or crystal growth conditions); it is assumed that the full experimental description of each phasing set would be recorded in a separate data block (see Example 3.6.6.6).

Data items in the PHASING_SET category identify each set of diffraction data used in a phasing experiment and can be used to summarize relevant experimental conditions. Because a given data set may be used in a number of different ways (for example, as an isomorphous derivative and as a component of a multiple-wavelength calculation), it is appropriate to store the reflections in a category distinct from either the PHASING_MAD or PHASING_MIR family of categories, but accessible to both these families (and any similar categories that might be introduced later to describe new phasing methods). Figs. 3.6.6.1 and 3.6.6.2 show how reference is made to the relevant sets from within the PHASING_MAD and PHASING_MIR categories.

Each phasing set is given a unique value of **_phasing_set.id**. The other PHASING_SET data items record the cell dimensions and

Example 3.6.6.6. *The phasing sets used in the structure determination of bovine plasma retinol-binding protein (Zanotti et al., 1993) described with data items in the* PHASING_SET *and* PHASING_SET_REFLN *categories.*

```
_phasing_set.id                    'NS1-96'
_phasing_set.cell_angle_alpha      90.0
_phasing_set.cell_angle_beta       90.0
_phasing_set.cell_angle_gamma      90.0
_phasing_set.cell_length_a         38.63
_phasing_set.cell_length_b         38.63
_phasing_set.cell_length_c         82.88
_phasing_set.radiation_wavelength  1.5145
_phasing_set.detector_type         'image plate'
_phasing_set.detector_specific     'RXII'

_loop
_phasing_set_refln.set_id
_phasing_set_refln.index_h
_phasing_set_refln.index_k
_phasing_set_refln.index_l
_phasing_set_refln.F_meas_au
_phasing_set_refln.F_meas_sigma_au
  'NS1-96'  15  15  32  181.79  3.72
  'NS1-96'  15  15  33   34.23  1.62
# - - - abbreviated - - -
```

angles associated with each phasing set, the wavelength of the radiation used in the experiment, the source of the radiation, the detector type, and the ambient temperature.

Data items in the PHASING_SET_REFLN category are used to record the values of the measured structure factors and their uncertainties. Several distinct data sets may be present in this list, with reflections in each set identified by the appropriate value of **_phasing_set_refln.set_id**.

3.6.6.2. Refinement

The categories describing refinement are as follows:

REFINE group

Overall description of the refinement (§3.6.6.2.1)

REFINE

REFINE_FUNCT_MINIMIZED

Analysis of the refined structure (§3.6.6.2.2)

REFINE_ANALYZE

Restraints and refinement by shells of resolution (§3.6.6.2.3)

REFINE_LS_RESTR

REFINE_LS_RESTR_NCS

REFINE_LS_RESTR_TYPE

REFINE_LS_SHELL

REFINE_LS_CLASS

Equivalent atoms in the refinement (§3.6.6.2.4)

REFINE_B_ISO

REFINE_OCCUPANCY

History of the refinement (§3.6.6.2.5)

REFINE_HIST

The macromolecular CIF dictionary contains many more data items for describing the refinement process than the core CIF dictionary does. In addition to new items in the REFINE category itself, additional categories have been introduced to describe in great detail the function minimized and the restraints applied, and the history of the refinement process, which often has many cycles. The REFINE_ANALYZE category can be used to give details of many of the quantities that may be used to assess the quality of the refinement. The REFINE_LS_SHELL category allows results to be reported by shells of resolution, and in effect replaces the more general core CIF category REFINE_LS_CLASS.

3.6.6.2.1. *Overall description of the refinement*

The data items in these categories are as follows:

(*a*) REFINE

- **_refine.entry_id**
  → **_entry.id**
  _refine.aniso_B[1][1]
  _refine.aniso_B[1][2]
  _refine.aniso_B[1][3]
  _refine.aniso_B[2][2]
  _refine.aniso_B[2][3]
  _refine.aniso_B[3][3]
  _refine.B_iso_max
  _refine.B_iso_mean
  _refine.B_iso_min
  _refine.correlation_coeff_Fo_to_Fc
  _refine.correlation_coeff_Fo_to_Fc_free
  _refine.details (∼ ***_refine_special_details***)
+ ***_refine.diff_density_max***
+ ***_refine.diff_density_min***
+ ***_refine.diff_density_rms***
  _refine.ls_abs_structure_details
+ ***_refine.ls_abs_structure_Flack***
+ ***_refine.ls_abs_structure_Rogers***
  _refine.ls_d_res_high
  _refine.ls_d_res_low
+ ***_refine.ls_extinction_coef***
  _refine.ls_extinction_expression
  _refine.ls_extinction_method
+ ***_refine.ls_goodness_of_fit_all***
  _refine.ls_goodness_of_fit_gt
+ ***_refine.ls_goodness_of_fit_obs***
  _refine.ls_goodness_of_fit_ref
  _refine.ls_hydrogen_treatment
  _refine.ls_matrix_type
  _refine.ls_number_constraints
  _refine.ls_number_parameters
  _refine.ls_number_reflns_all
  _refine.ls_number_reflns_obs
  (∼ ***_refine_ls_number_reflns***)
  _refine.ls_number_reflns_R_free
  _refine.ls_number_reflns_R_work
  _refine.ls_number_restraints
  _refine.ls_percent_reflns_obs
  _refine.ls_percent_reflns_R_free
  _refine.ls_R_factor_all
  _refine.ls_R_factor_gt
  _refine.ls_R_factor_obs
  _refine.ls_R_factor_R_free
  _refine.ls_R_factor_R_free_error
  _refine.ls_R_factor_R_free_error_details
  _refine.ls_R_factor_R_work
  _refine.ls_R_Fsqd_factor_obs
  (∼ ***_refine_ls_R_Fsqd_factor***)
  _refine.ls_R_I_factor_obs (∼ ***_refine_ls_R_I_factor***)
  _refine.ls_redundancy_reflns_all
  _refine.ls_redundancy_reflns_obs
  _refine.ls_restrained_S_all
  _refine.ls_restrained_S_obs
  _refine.ls_shift_over_esd_max
  (∼ ***_refine_ls_shift/esd_max***)
  _refine.ls_shift_over_esd_mean
  (∼ ***_refine_ls_shift/esd_mean***)
  _refine.ls_shift_over_su_max
  (∼ ***_refine_ls_shift/su_max***)
  _refine.ls_shift_over_su_max_lt
  (∼ ***_refine_ls_shift/su_max_lt***)
  _refine.ls_shift_over_su_mean
  (∼ ***_refine_ls_shift/su_mean***)
  _refine.ls_shift_over_su_mean_lt
  (∼ ***_refine_ls_shift/su_mean_lt***)
  _refine.ls_structure_factor_coef
  _refine.ls_weighting_details
  _refine.ls_weighting_scheme
  _refine.ls_wR_factor_all
  _refine.ls_wR_factor_obs
  _refine.ls_wR_factor_R_free
  _refine.ls_wR_factor_R_work
  _refine.occupancy_max
  _refine.occupancy_min
  _refine.overall_FOM_free_R_set
  _refine.overall_FOM_work_R_set
  _refine.overall_SU_B
  _refine.overall_SU_ML
  _refine.overall_SU_R_Cruickshank_DPI
  _refine.overall_SU_R_free
  _refine.solvent_model_details
  _refine.solvent_model_param_bsol
  _refine.solvent_model_param_ksol

(*b*) REFINE_FUNCT_MINIMIZED

- **_refine_funct_minimized.type**
  _refine_funct_minimized.number_terms
  _refine_funct_minimized.residual
  _refine_funct_minimized.weight

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ∼ symbol. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string **_esd** *to the data name listed.*

There is already an extensive set of data names in the REFINE category of the core dictionary, and Section 3.2.3.1 should be read with the present section. The only data items discussed in this section are entries in the mmCIF dictionary that do not have a counterpart in the core CIF dictionary. Analogues of a number of *R* factors in the core CIF dictionary have been added to the mmCIF dictionary to express these same *R* factors indepen-

dently for the free and working sets of reflections. The remaining new data items have more specialized roles, which are discussed below.

The data item **_refine.entry_id** has been added to the REFINE category to provide the formal category key required by the DDL2 data model.

Many macromolecular structure refinements now use the statistical cross-validation technique of monitoring a 'free' R factor (Brünger, 1997). $R_{\rm free}$ is calculated the same way as the conventional least-squares R factor, but using a small subset of reflections that are not used in the refinement of the structural model. Thus $R_{\rm free}$ tests how well the model predicts experimental observations that are not themselves used to fit the model.

The mmCIF dictionary provides data names for $R_{\rm free}$ and for the complementary $R_{\rm work}$ values for the 'working' set of reflections, which are the reflections that are used in the refinement. Separate data items are provided for unweighted and weighted versions of each R factor. A fixed percentage of the total number of reflections is usually assigned to the free group, and this percentage can be specified. Further details about the method used for selecting the free reflections can be given using **_reflns.R_free_details**. The estimated error in the $R_{\rm free}$ value may also be given, along with the method used for determining its value.

The purposes of having a set of reflections that are not used in the refinement are to monitor the progress of the refinement and to ensure that the R factor is not being artificially reduced by the introduction of too many parameters. However, as the refinement converges, the working and free R factors both approach stable values. It is common practice, particularly in structures at high resolution, to stop monitoring $R_{\rm free}$ at this point and to include all the reflections in the final rounds of refinement. It is thus worth noting a distinction between **_refine.ls_R_factor_obs** and **_refine.ls_R_factor_R_work**: **_refine.ls_R_factor_obs** relates to a refinement in which all reflections more intense than a specified threshold were used, while **_refine.ls_R_factor_R_work** relates to a refinement in which a subset of the observed reflections were excluded from the refinement and were used to calculate the free R factor. The dictionary allows the use of both values if a free R factor were calculated for most of the refinement, but all of the observed reflections were used in the final rounds of refinement; the protocol for this may be explained in **_refine.details**. When a full history of the refinement is provided using data items in the REFINE_HIST category, it is preferable to specify a change in protocol using data items in this category.

Other data items help to provide an assessment of the quality of the refinement. The scale-independent correlation coefficient between the observed and calculated structure factors may be recorded for the reflections included in the refinement using the data item **_refine.correlation_coeff_Fo_to_Fc**. There is a similar data item for the reflections that were not included in the refinement.

Overall standard uncertainties for positional and displacement parameters can be recorded according to a number of conventions. A maximum-likelihood residual for the positional parameters can be given using **_refine.overall_SU_ML** and the corresponding value for the displacement parameters can be given using **_refine.overall_SU_B**. Diffraction-component precision indexes for the displacement parameters based on the crystallographic R factor (the Cruickshank DPI; Cruickshank, 1999) can be given using **_refine.overall_SU_R_Cruickshank_DPI**. The corresponding value for $R_{\rm free}$ can be given using **_refine.overall_SU_R_free**.

Example 3.6.6.7. *Results of the overall refinement of an HIV-1 protease structure (PDB 5HVP) described using data items in the* REFINE *and* REFINE_FUNCT_MINIMIZED *categories.*

```
_refine.entry_id                          '5HVP'
_refine.ls_number_reflns_obs              12901
_refine.ls_number_restraints              6609
_refine.ls_number_parameters              7032
_refine.ls_R_factor_obs                   0.176
_refine.ls_weighting_scheme               calc
_refine.ls_weighting_details
;  Sigdel model of Konnert-Hendrickson:
   Sigdel: Afsig +  Bfsig*(sin(theta)/lambda-1/6)
   Afsig = 22.0, Bfsig = -150.0 at the beginning
      of refinement.
   Afsig = 15.5, Bfsig =  -50.0 at the end of
      refinement.
;
loop_
  _refine_funct_minimized.type
  _refine_funct_minimized.number_terms
  _refine_funct_minimized.residual
    'sum(W*Delta(Amplitude)^2^'       3009   1621.3
    'sum(W*Delta(Plane+Rigid)^2^'       85     56.68
    'sum(W*Delta(Distance)^2^'        1219    163.59
    'sum(W*Delta(U-tempfactors)^2^'   1192     69.338
```

The quality of a data set used for the refinement of a macromolecular structure is often given not only in terms of the scaling residuals, but also in terms of the data redundancy (the ratio of the number of reflections measured to the number of crystallographically unique reflections). Data items are provided to express the redundancy of all reflections, as well as those that have been marked as 'observed' (*i.e.* exceeding the threshold for inclusion in the refinement). The percentage of the total number of reflections that are considered observed is another metric of the quality of the data set, and a data item is provided for this (**_refine.ls_percent_reflns_obs**).

The limited resolution of many macromolecular data sets makes it inappropriate to refine anisotropic displacement factors for each atom. For these low- to medium-resolution studies, an overall anisotropic displacement model may be refined. The data items **_refine.aniso_B*** are provided for recording the unique elements of the matrix that describes the refined anisotropy.

The two-parameter method for modelling the contribution of the bulk solvent to the scattering proposed by Tronrud is used in several refinement programs. The data items **_refine.solvent_model_*** can be used to record the scale and displacement factors of this model, and any special aspects of its application to the refinement.

The average phasing figure of merit can be given for the working and free reflections. Unusually high or low values of displacement factors or occupancies can be a sign of problems with the refinement, so data items are provided to record the high, low and mean values of each. Further indicators of the quality of the refinement are found in the REFINE_ANALYZE category (Section 3.6.6.2.2).

The data items in the REFINE_FUNCT_MINIMIZED category allow a brief description of the function minimized during refinement to be given (Example 3.6.6.7). It is not possible to reconstruct the functioned minimized during the refinement by automatic parsing of the values of these data items, but the details given in them may still be helpful to someone reading the mmCIF.

3.6.6.2.2. *Analysis of the refined structure*

The data items in this category are as follows:

REFINE_ANALYZE

- **_refine_analyze.entry_id**
  → **_entry.id**

```
_refine_analyze.Luzzati_coordinate_error_free
_refine_analyze.Luzzati_coordinate_error_obs
_refine_analyze.Luzzati_d_res_low_free
_refine_analyze.Luzzati_d_res_low_obs
_refine_analyze.Luzzati_sigma_a_free
_refine_analyze.Luzzati_sigma_a_free_details
_refine_analyze.Luzzati_sigma_a_obs
_refine_analyze.Luzzati_sigma_a_obs_details
_refine_analyze.number_disordered_residues
_refine_analyze.occupancy_sum_hydrogen
_refine_analyze.occupancy_sum_non_hydrogen
_refine_analyze.RG_d_res_high
_refine_analyze.RG_d_res_low
_refine_analyze.RG_free
_refine_analyze.RG_free_work_ratio
_refine_analyze.RG_work
```

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

In small-molecule crystallography, there is general agreement on the metrics that should be used to assess the quality of a structure determination, and data items in the REFINE category of the core CIF dictionary can be used to record them. For macromolecular structure determinations, no such agreement has been achieved yet and new metrics are frequently suggested as the field evolves. The REFINE_ANALYZE category can be used to record the metrics that were in common use at the time that the mmCIF dictionary was constructed; it is anticipated that new metrics will be added in future versions of the dictionary, and that some of the current metrics may fall into disuse.

Luzzati (1952) devised a method for estimating the average positional shift that would be needed in an idealized refinement to reach an R factor of zero by using a plot of R factors against resolution. For some time, macromolecular crystallographers have used a modification of this approach to assess the average positional error. Recent practice has used Luzzati plots based on the free R values to yield a cross-validated error estimate. Data items are provided for recording these coordinate-error estimates and the range of resolution included in the plot (Example 3.6.6.8). Related data names allow the specification of the value of σ_a used in constructing the Luzzati plot.

A general feature of introducing more parameters in the model of the structure is a reduction in the R factor, but the statistical significance of this is often obscured by the simultaneous reduction in the ratio of observations to parameters. Attempts to extend Hamilton's (1965) test to macromolecular structures are usually confounded by the use of restraints. Tickle *et al.* (1998) proposed the use of a Hamilton generalized R factor analyzed separately for reflections in the working set (those used in the refinement) and for reflections in the free set (those set aside for cross validation), and these metrics are often reported in the literature. Data items are provided for recording the Hamilton generalized R factor for the working and free set of reflections, and for the ratio of the two.

Other indicators of a successful refinement involve the relative order of the model. Data items are provided for recording the sum of the occupancies of the hydrogen and non-hydrogen atoms in the model. The number of disordered residues may also be recorded.

3.6.6.2.3. *Restraints and refinement by shells of resolution*

The data items in these categories are as follows:

(*a*) REFINE_LS_RESTR

```
• _refine_ls_restr.type
  _refine_ls_restr.criterion
  _refine_ls_restr.dev_ideal
  _refine_ls_restr.dev_ideal_target
  _refine_ls_restr.number
  _refine_ls_restr.rejects
  _refine_ls_restr.weight
```

Example 3.6.6.8. *Aspects of the refinement of an HIV-1 protease structure (PDB 5HVP) described with data items in the REFINE_ANALYZE category.*

```
loop_
_refine_analyze.entry_id                          '5HVP'
_refine_analyze.Luzzati_coordinate_error_obs       0.32
_refine_analyze.Luzzati_d_res_low_obs              5.0
```

(*b*) REFINE_LS_RESTR_NCS

```
• _refine_ls_restr_ncs.dom_id
        → _struct_ncs_dom.id
  _refine_ls_restr_ncs.ncs_model_details
  _refine_ls_restr_ncs.rms_dev_B_iso
  _refine_ls_restr_ncs.rms_dev_position
  _refine_ls_restr_ncs.weight_B_iso
  _refine_ls_restr_ncs.weight_position
```

(*c*) REFINE_LS_RESTR_TYPE

```
• _refine_ls_restr_type.type
        → _refine_ls_restr.type
  _refine_ls_restr_type.distance_cutoff_high
  _refine_ls_restr_type.distance_cutoff_low
```

(*d*) REFINE_LS_SHELL

```
• _refine_ls_shell.d_res_high
• _refine_ls_shell.d_res_low
  _refine_ls_shell.number_reflns_all
  _refine_ls_shell.number_reflns_obs
  _refine_ls_shell.number_reflns_R_free
  _refine_ls_shell.number_reflns_R_work
  _refine_ls_shell.percent_reflns_obs
  _refine_ls_shell.percent_reflns_R_free
  _refine_ls_shell.R_factor_all
  _refine_ls_shell.R_factor_obs
  _refine_ls_shell.R_factor_R_free
  _refine_ls_shell.R_factor_R_free_error
  _refine_ls_shell.R_factor_R_work
  _refine_ls_shell.redundancy_reflns_all
  _refine_ls_shell.redundancy_reflns_obs
  _refine_ls_shell.wR_factor_all
  _refine_ls_shell.wR_factor_obs
  _refine_ls_shell.wR_factor_R_free
  _refine_ls_shell.wR_factor_R_work
```

(*e*) REFINE_LS_CLASS

```
• _refine_ls_class.code
  _refine_ls_class.d_res_high
  _refine_ls_class.d_res_low
  _refine_ls_class.R_factor_all
  _refine_ls_class.R_factor_gt
  _refine_ls_class.R_Fsqd_factor
  _refine_ls_class.R_I_factor
  _refine_ls_class.wR_factor_all
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

These categories were introduced in the mmCIF dictionary to allow a detailed description of several aspects of structure refinement to be given. Data items in the REFINE_LS_RESTR category allow geometric restraints to be specified and the deviations of restrained parameters from ideal values in the final model to be given. The type of the geometric restraints can be described in more detail using data items in the REFINE_LS_RESTR_TYPE category. Data items in the REFINE_LS_RESTR_NCS category can be used to give information about any restraints on noncrystallographic symmetry used in the refinement and the category REFINE_LS_SHELL contains data items that allow the results of refinement to be given by shells of resolution.

Data items in the REFINE_LS_RESTR category can be used to record details about the restraints applied to various classes of parameters during least-squares refinement (Example 3.6.6.9). It is clearly useful to tabulate the various classes of restraint, their deviation from ideal target values and the criteria used to reject

Example 3.6.6.9. *Results of the refinement of an HIV-1 protease structure (PDB 5HVP) described with data items in the* REFINE_LS_RESTR *and* REFINE_LS_SHELL *categories.*

```
loop_
_refine_ls_restr.type
_refine_ls_restr.dev_ideal_target
_refine_ls_restr.dev_ideal
_refine_ls_restr.number
_refine_ls_restr.criterion
_refine_ls_restr.rejects
'p_bond_d'          0.020  0.018  1654 '>2 sigma' 22
'p_angle_d'         0.030  0.038  2246 '>2 sigma' 139
'p_planar_d'        0.040  0.043  498  '>2 sigma' 21
'p_planar'          0.020  0.015  270  '>2 sigma' 1
'p_chiral'          0.150  0.177  278  '>2 sigma' 2
'p_singtor_nbd'     0.500  0.216  582  '>2 sigma' 0
'p_multtor_nbd'     0.500  0.207  419  '>2 sigma' 0
'p_xyhbond_nbd'     0.500  0.245  149  '>2 sigma' 0
'p_planar_tor'      3.0    2.6    203  '>2 sigma' 9
'p_staggered_tor'   15.0   17.4   298  '>2 sigma' 31
'p_orthonormal_tor' 20.0   18.1   12   '>2 sigma' 1

loop_
_refine_ls_shell.d_res_low
_refine_ls_shell.d_res_high
_refine_ls_shell.number_reflns_obs
_refine_ls_shell.R_factor_obs
   8.00   4.51   1226   0.196
   4.51   3.48   1679   0.146
   3.48   2.94   2014   0.160
   2.94   2.59   2147   0.182
   2.59   2.34   2127   0.193
   2.34   2.15   2061   0.203
   2.15   2.00   1647   0.188
```

parameters that lie too far from a target, as these data are often published as part of a description of the refinement and are often deposited with the coordinates in an archive. However, the types of restraints applied depend strongly on the software package used, and as new refinement packages regularly become available, it was clearly not advisable to provide program-specific data items in the mmCIF dictionary. The approach taken in the mmCIF dictionary has been to allow the value of **_refine_ls_restr.type** to be a free-text field, so that arbitrary labels can be given to restraints that are particular to a software package, but to recommend the use of specific labels for restraints applied by particular programs. The dictionary provides examples for labels specific to the programs *PROTIN/PROLSQ* (Hendrickson & Konnert, 1979) and *RESTRAIN* (Driessen *et al.*, 1989). These program-specific representations have particular prefixes; thus the value `p_bond_d` is a bond-distance restraint as applied by *PROTIN/PROLSQ*. Values for **_refine_ls_restr.type** appropriate for other refinement programs may be suggested in future versions of the mmCIF dictionary.

Data items in the REFINE_LS_RESTR_TYPE category can be used to specify the ranges within which quantities are allowed to vary for each type of restraint. The special value indicated by a full stop (.) represents a restraint unbounded on the high or low side.

Data items in the REFINE_LS_RESTR_NCS category can be used to record details about the restraints applied to atom positions in domains related by noncrystallographic symmetry during least-squares refinement, and also to record the deviation of the restrained atomic parameters at the end of the refinement. The domains related by noncrystallographic symmetry are defined in the STRUCT_NCS_DOM and related categories (see Section 3.6.7.5.5). The quantities that can be recorded for each restrained domain are the root-mean-square deviations of the displacement and positional parameters, and the weighting coefficients used in the noncrystallographic restraint of each type of parameter. Any special aspects of the way the restraints were applied may be described using **_refine_ls_restr_ncs.ncs_model_details**.

Data items in the REFINE_LS_SHELL category are used to summarize details of the results of the least-squares refinement by shells of resolution (Example 3.6.6.9). The resolution range, in ångströms, forms the category key; for each shell the quantities reported, such as the number of reflections above the threshold for counting as significantly intense, are all defined in the same way as the corresponding data items used to describe the results of the overall refinement in the REFINE category.

The core dictionary category REFINE_LS_CLASS was introduced after the release of the first version of the mmCIF dictionary. It provides a more general way of describing the treatment of particular subsets of the observations, but it is not expected to be used in macromolecular structural studies, where partition by shells of resolution is traditional.

3.6.6.2.4. *Equivalent atoms in the refinement*

The data items in these categories are as follows:

(*a*) REFINE_B_ISO

- **_refine_B_iso.class**
  _refine_B_iso.details
  _refine_B_iso.treatment
  _refine_B_iso.value

(*b*) REFINE_OCCUPANCY

- **_refine_occupancy.class**
  _refine_occupancy.details
  _refine_occupancy.treatment
  _refine_occupancy.value

The bullet (•) *indicates a category key.*

In macromolecular structure refinement, displacement factors or occupancies are often treated as equivalent for groups of atoms. An example would be the case where most of the atoms in the structure are refined with isotropic displacement factors, but a bound metal atom is allowed to refine anisotropically. Another example would be where the occupancies for all of the atoms in the protein part of a macromolecular complex are fixed at 1.0, but the occupancies of atoms in a bound inhibitor are refined. The REFINE_B_ISO and REFINE_OCCUPANCY categories can be used to record this information (Example 3.6.6.10).

Example 3.6.6.10. *The handling of displacement factors and occupancies during the refinement of an HIV-1 protease structure (PDB 5HVP) described with data items in the* REFINE_B_ISO *and* REFINE_OCCUPANCY *categories.*

```
loop_
_refine_B_iso.class
_refine_B_iso.treatment
   'protein'    isotropic
   'solvent'    isotropic
   'inhibitor'  isotropic

loop_
_refine_occupancy.class
_refine_occupancy.treatment
_refine_occupancy.value
_refine_occupancy.details
   'protein'                   fix  1.00  .
   'solvent'                   fix  1.00  .
   'inhibitor orientation 1'   fix  0.65  .
   'inhibitor orientation 2'   fix  0.35
; The inhibitor binds to the enzyme in two
  alternative conformations. The occupancy of
  each conformation was adjusted so as to result
  in approximately equal mean thermal factors
  for the atoms in each conformation.
;
```

Example 3.6.6.11. *An example of one cycle of refinement described with data items in the* REFINE_HIST *category.*

```
_refine_hist.cycle_id                C134
_refine_hist.d_res_high              1.85
_refine_hist.d_res_low               20.0
_refine_hist.number_atoms_solvent     217
_refine_hist.number_atoms_total       808
_refine_hist.number_reflns_all       6174
_refine_hist.number_reflns_obs       4886
_refine_hist.number_reflns_R_free     476
_refine_hist.number_reflns_R_work    4410
_refine_hist.R_factor_all            .265
_refine_hist.R_factor_obs            .195
_refine_hist.R_factor_R_free         .274
_refine_hist.R_factor_R_work         .160
_refine_hist.details
; Add majority of solvent molecules. B factors
  refined by group. Continued to remove
  misplaced water molecules.
;
```

Data items in the REFINE_B_ISO category can be used to record details of the treatment of isotropic *B* (displacement) factors during refinement. There is no formal link between the classes identified by `_refine_B_iso.class` and individual atom sites, although relationships may be inferred if the class names are carefully chosen. The category allows the treatment of the atoms in each class (isotropic, anisotropic or fixed) and the value assigned for fixed isotropic *B* factors to be recorded. Any special details can be given in a free-text field.

Data items in the REFINE_OCCUPANCY category can be used to record details of the treatment of occupancies of groups of atom sites during refinement. As with the treatment of displacement factors in the REFINE_B_ISO category, the classes itemized by `_refine_occupancy.class` are not formally linked to the individual atom sites, but the relationships may be deduced if the class names are chosen carefully.

3.6.6.2.5. *History of the refinement*

The data items in this category are as follows:

REFINE_HIST

- `_refine_hist.cycle_id`
  `_refine_hist.details`
  `_refine_hist.d_res_high`
  `_refine_hist.d_res_low`
  `_refine_hist.number_atoms_solvent`
  `_refine_hist.number_atoms_total`
  `_refine_hist.number_reflns_all`
  `_refine_hist.number_reflns_obs`
  `_refine_hist.number_reflns_R_free`
  `_refine_hist.number_reflns_R_work`
  `_refine_hist.R_factor_all`
  `_refine_hist.R_factor_obs`
  `_refine_hist.R_factor_R_free`
  `_refine_hist.R_factor_R_work`

The bullet (•) *indicates a category key.*

Data items in the REFINE_HIST category can be used to record details about the various steps in the refinement of the structure. They do not provide as thorough a description of the refinement as can be given in other categories for the final model, but instead allow a summary of the progress of the refinement to be given and supported by a small set of representative statistics.

The category is sufficiently compact that a large number of cycles could be summarized, but it is not expected that every cycle of refinement would be routinely reported. Example 3.6.6.11 shows an entry for a single cycle of refinement. It is likely that an author would present a representative sequence of entries in a looped list.

3.6.6.3. Reflection measurements

The categories describing the reflections used in the refinement are as follows:

REFLN group
Individual reflections (§3.6.6.3.1)
  REFLN
  REFLN_SYS_ABS
Groups of reflections (§3.6.6.3.2)
  REFLNS
  REFLNS_SCALE
  REFLNS_SHELL
  REFLNS_CLASS

Data items in the REFLN category can be used to give information about the individual reflections that were used to derive the final model. The related category REFLN_SYS_ABS allows the reflections that should be systematically absent for the space group in which the structure was refined to be tabulated. Data items in the REFLNS category can be used to record information that applies to all of the reflections. Scale factors can be listed in the REFLNS_SCALE category, while the data items in REFLNS_SHELL can be used to record information about the reflection set by shells of resolution. The core CIF dictionary category REFLNS_CLASS, which can be used for a general classification of reflection groups according to criteria other than resolution shell, is not expected to be used in mmCIF applications.

3.6.6.3.1. *Individual reflections*

The data items in these categories are as follows:

(*a*) REFLN

- `_refln.index_h`
- `_refln.index_k`
- `_refln.index_l`

  `_refln.A_calc`
  `_refln.A_calc_au`
  `_refln.A_meas`
  `_refln.A_meas_au`
  `_refln.B_calc`
  `_refln.B_calc_au`
  `_refln.B_meas`
  `_refln.B_meas_au`
  `_refln.class_code`
  `_refln.crystal_id`
  → `_exptl_crystal.id`
  `_refln.d_spacing`
  `_refln.F_calc`
  `_refln.F_calc_au`
  `_refln.F_meas`
  `_refln.F_meas_au`
  `_refln.F_meas_sigma` (∼ `_refln_F_sigma`)
  `_refln.F_meas_sigma_au`
  `_refln.F_squared_calc`
  `_refln.F_squared_meas`
  `_refln.F_squared_sigma`
  `_refln.fom`
  `_refln.include_status`
  `_refln.intensity_calc`
  `_refln.intensity_meas`
  `_refln.intensity_sigma`
  `_refln.mean_path_length_tbar`
  `_refln.phase_calc`
  `_refln.phase_meas`
  `_refln.refinement_status`
  `_refln.scale_group_code`
  → `_reflns_scale.group_code`
  `_refln.sint_over_lambda` (∼ `_refln_sint/lambda`)
  `_refln.status` (∼ `_refln_observed_status`)
  `_refln.symmetry_epsilon`
  `_refln.symmetry_multiplicity`
  `_refln.wavelength`

Example 3.6.6.12. *Part of the reflection list for an HIV-1 protease structure (PDB 5HVP) described with data items in the* REFLN *category.*

```
loop_
_refln.index_h
_refln.index_k
_refln.index_l
_refln.F_squared_calc
_refln.F_squared_meas
_refln.F_squared_sigma
_refln.status
   2   0   0       85.57       58.90      1.45 o
   3   0   0    15718.18    15631.06     30.40 o
   4   0   0    55613.11    49840.09     61.86 o
   5   0   0      246.85      241.86     10.02 o
   6   0   0       82.16       69.97      1.93 o
   7   0   0     1133.62      947.79     11.78 o
   8   0   0     2558.04     2453.33     20.44 o
   9   0   0      283.88      393.66      7.79 o
  10   0   0      283.70      171.98      4.26 o
```

```
  _refln.wavelength_id
        →  _diffrn_radiation.wavelength_id
```

(*b*) REFLN_SYS_ABS
- **_refln_sys_abs.index_h**
- **_refln_sys_abs.index_k**
- **_refln_sys_abs.index_l**

  _refln_sys_abs.I
  _refln_sys_abs.I_over_sigmaI
  _refln_sys_abs.sigmaI

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (**.**) *to an underscore* (_) *except where indicated by the* ∼ *symbol.*

Data items in the REFLN category are used in the same way in the mmCIF and core CIF dictionaries, and Section 3.2.3.2.1 can be consulted for details. However, in macromolecular crystallography it is not usual for reflection intensities to be given in units of electrons (the units specified by the core CIF dictionary). Thus it was necessary to introduce in the mmCIF dictionary data items for the magnitudes of structure factors and their *A* and *B* components in arbitrary units (Example 3.6.6.12). A figure of merit (**_refln.fom**) can also be included for reflections that were phased using experimental methods.

The REFLN_SYS_ABS category allows the intensities of the reflections that should be systematically absent to be tabulated. The ratio of the intensity to its standard uncertainty, given in the data item **_refln_sys_abs.I_over_sigmaI**, can be used to assess whether the reflection is indeed absent. The decision as to whether it is absent is left to the user of the mmCIF and is not recorded in the mmCIF.

3.6.6.3.2. *Groups of reflections*

The data items in these categories are as follows:

(*a*) REFLNS
- **_reflns.entry_id**
  _reflns.B_iso_Wilson_estimate
  _reflns.data_reduction_details
  _reflns.data_reduction_method
  _reflns.d_resolution_high
  _reflns.d_resolution_low
  _reflns.details (∼ ***_reflns_special_details***)
  _reflns.Friedel_coverage
  _reflns.limit_h_max
  _reflns.limit_h_min
  _reflns.limit_k_max
  _reflns.limit_k_min
  _reflns.limit_l_max

  _reflns.limit_l_min
  _reflns.number_all (∼ ***_reflns_number_total***)
  _reflns.number_gt
  _reflns.number_obs (∼ ***_reflns_number_observed***)
  _reflns.observed_criterion
  _reflns.observed_criterion_F_max
  _reflns.observed_criterion_F_min
  _reflns.observed_criterion_I_max
  _reflns.observed_criterion_I_min
  _reflns.observed_criterion_sigma_F
  _reflns.observed_criterion_sigma_I
  _reflns.percent_possible_obs
  _reflns.R_free_details
  _reflns.Rmerge_F_all
  _reflns.Rmerge_F_obs
  _reflns.threshold_expression

(*b*) REFLNS_SCALE
- ***_reflns_scale.group_code***

  _reflns_scale.meas_F
  _reflns_scale.meas_F_squared
  _reflns_scale.meas_intensity

(*c*) REFLNS_SHELL
- ***_reflns_shell.d_res_high***
- ***_reflns_shell.d_res_low***

  _reflns_shell.meanI_over_sigI_all
  _reflns_shell.meanI_over_sigI_gt
  _reflns_shell.meanI_over_sigI_obs
  _reflns_shell.meanI_over_uI_all
  _reflns_shell.meanI_over_uI_gt
  _reflns_shell.number_measured_all
  _reflns_shell.number_measured_gt
  _reflns_shell.number_measured_obs
  _reflns_shell.number_possible
  _reflns_shell.number_unique_all
  _reflns_shell.number_unique_gt
  _reflns_shell.number_unique_obs
  _reflns_shell.percent_possible_all
  _reflns_shell.percent_possible_gt
  _reflns_shell.percent_possible_obs
  _reflns_shell.Rmerge_F_all
  _reflns_shell.Rmerge_F_gt
  _reflns_shell.Rmerge_F_obs
  _reflns_shell.Rmerge_I_all
  _reflns_shell.Rmerge_I_gt
  _reflns_shell.Rmerge_I_obs

(*d*) REFLNS_CLASS
- ***_reflns_class.code***

  _reflns_class.d_res_high
  _reflns_class.d_res_low
  _reflns_class.description
  _reflns_class.number_gt
  _reflns_class.number_total
  _reflns_class.R_factor_all
  _reflns_class.R_factor_gt
  _reflns_class.R_Fsqd_factor
  _reflns_class.R_I_factor
  _reflns_class.wR_factor_all

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (**.**) *to an underscore* (_) *except where indicated by the* ∼ *symbol.*

Data items in the REFLNS category of the core CIF dictionary can be used to summarize the properties or attributes of the complete set of reflections used in refinement (Section 3.2.3.2.2). The mmCIF dictionary adds a number of data items to this category, including the formal category key required by the DDL2 data model. There are also data items for describing the data-reduction method and recording any relevant details about data reduction, and for giving an estimate of the overall Wilson *B* factor for the data set.

A number of the new data items relate to the issue of how reflections are flagged as being observed and are thus used in the refinement. In the core CIF dictionary, the criteria used to consider

Example 3.6.6.13. *The data set used in the refinement of an HIV-1 protease structure (PDB 5HVP) described using data items in the* REFLNS *and* REFLNS_SHELL *categories.*

```
_reflns.entry_id                         '5HVP'
_reflns.data_reduction_method
; Xengen program scalei. Anomalous pairs were merged.
  Scaling proceeded in several passes, beginning with
  1-parameter fit and ending with 3-parameter fit.
;
_reflns.data_reduction_details
; Merging and scaling based on only those reflections
  with I > sigma(I).
;

_reflns.d_resolution_high                2.00
_reflns.d_resolution_low                 8.00

_reflns.limit_h_max                      22
_reflns.limit_h_min                      0
_reflns.limit_k_max                      46
_reflns.limit_k_min                      0
_reflns.limit_l_max                      57
_reflns.limit_l_min                      0

_reflns.number_obs                       7228
_reflns.observed_criterion_sigma_I       1.0
_reflns.details                          none

loop_
_reflns_shell.d_res_high
_reflns_shell.d_res_low
_reflns_shell.meanI_over_sigI_obs
_reflns_shell.number_measured_obs
_reflns_shell.number_unique_obs
_reflns_shell.percent_possible_obs
_reflns_shell.Rmerge_F_obs
   31.38   3.82   69.8   9024   2540   96.8    1.98
    3.82   3.03   26.1   7413   2364   95.1    3.85
    3.03   2.65   10.5   5640   2123   86.2    6.37
    2.65   2.41    6.4   4322   1882   76.8    8.01
    2.41   2.23    4.3   3247   1714   70.4    9.86
    2.23   2.10    3.1   1140    812   33.3   13.99
```

a reflection as being observed are given using the data item `_reflns.observed_criterion`. This is a free-text field so is not automatically parsable. Therefore it is supplemented in the mmCIF dictionary by data items that can be used to stipulate the criterion in terms of the values of F, I or the uncertainties in these quantities (Example 3.6.6.13). The percentage of the total number of reflections that meet the criterion can be recorded.

Data items are also provided for describing the selection of the reflections used to calculate the free R factor, and for giving the $R_{\rm merge}$ values for all reflections and for the subset of 'observed' reflections. Data items in the REFLNS_SCALE and REFLNS_SHELL categories are used in the same way in the mmCIF and core CIF dictionaries, and Section 3.2.3.2.2 can be consulted for details.

As with the related categories DIFFRN_REFLNS_CLASS and REFINE_LS_CLASS, the core dictionary category REFLNS_CLASS was introduced after the release of the first version of the mmCIF dictionary. It provides a more general way of describing the treatment of particular subsets of the observations, but it is not expected to be used in macromolecular structural studies, where partition by shells of resolution is traditional.

3.6.7. Atomicity, chemistry and structure

The basic concepts of the mmCIF model for describing a macromolecular structure were outlined in Section 3.6.3. The present section describes the components of the model in more detail. The category groups used to describe the molecular chemistry and structure are: the ATOM group describing atom positions (Section 3.6.7.1); the CHEMICAL, CHEM_COMP and CHEM_LINK groups describing molecular chemistry (Section 3.6.7.2); the ENTITY group describing distinct chemical species (Section 3.6.7.3); the GEOM group describing molecular or packing geometry (Section 3.6.7.4); the STRUCT group describing the large-scale features of molecular structure (Section 3.6.7.5); and the SYMMETRY group describing the symmetry and space group (Section 3.6.7.6).

The CHEMICAL category group itself is not generally used in an mmCIF. The purpose of this category group in the core CIF dictionary is to specify the chemical identity and connectivity of the relatively simple molecular or ionic species in a small-molecule or inorganic crystal. In principle, a macromolecular structure determined to atomic resolution could be represented as a coherent chemical entity with a complete connectivity graph. However, in practice, biological macromolecules are built from units from a library of models of standard amino acids, nucleotides and sugars. Data items in the CHEM_COMP and CHEM_LINK category groups of the mmCIF dictionary describe the internal connectivity and standard bonding processes between these units.

Molecular or packing geometry is also rarely tabulated for large macromolecular complexes, so the GEOM category group is rarely used in an mmCIF.

3.6.7.1. Atom sites

The categories describing atom sites are as follows:

ATOM group
Individual atom sites (§3.6.7.1.1)
  ATOM_SITE
  ATOM_SITE_ANISOTROP
Collections of atom sites (§3.6.7.1.2)
  ATOM_SITES
  ATOM_SITES_FOOTNOTE
Atom types (§3.6.7.1.3)
  ATOM_TYPE
Alternative conformations (§3.6.7.1.4)
  ATOM_SITES_ALT
  ATOM_SITES_ALT_ENS
  ATOM_SITES_ALT_GEN

The ATOM category group represents a compromise between the representation of a small-molecule structure as an annotated list of atomic coordinates and the need in macromolecular crystallography to present a more structured view organized around residues, chains, sheets, turns, helices *etc.* The locations of individual atoms and other information about the atom sites are given using data items in this category group. The categories within the group may be classified as shown in the summary above.

The ATOM_SITE, ATOM_SITES and ATOM_TYPE categories have many data items that are aliases of equivalent data items in the same categories in the core CIF dictionary, but the conventions for the labelling of the atom sites are different.

The ATOM_SITE_ANISOTROP and ATOM_SITES_FOOTNOTE categories are new to the mmCIF dictionary, as are the categories related to alternative conformations: ATOM_SITES_ALT, ATOM_SITES_ALT_ENS and ATOM_SITES_ALT_GEN.

3.6.7.1.1. *Individual atom sites*

The data items in these categories are as follows:

(*a*) ATOM_SITE
• **`_atom_site.id`** (∼ ***`_atom_site_label`***)
  `_atom_site.adp_type`
\+ **`_atom_site.aniso_B[1][1]`**
  ⇌ ***`_atom_site_anisotrop.B[1][1]`***

+ _atom_site.aniso_B[1][2]
  ⇌ _atom_site_anisotrop.B[1][2]
+ _atom_site.aniso_B[1][3]
  ⇌ _atom_site_anisotrop.B[1][3]
+ _atom_site.aniso_B[2][2]
  ⇌ _atom_site_anisotrop.B[2][2]
+ _atom_site.aniso_B[2][3]
  ⇌ _atom_site_anisotrop.B[2][3]
+ _atom_site.aniso_B[3][3]
  ⇌ _atom_site_anisotrop.B[3][3]
  _atom_site.aniso_ratio
  ⇌ _atom_site_anisotrop.ratio
+ _atom_site.aniso_U[1][1]
  ⇌ _atom_site_anisotrop.U[1][1]
+ _atom_site.aniso_U[1][2]
  ⇌ _atom_site_anisotrop.U[1][2]
+ _atom_site.aniso_U[1][3]
  ⇌ _atom_site_anisotrop.U[1][3]
+ _atom_site.aniso_U[2][2]
  ⇌ _atom_site_anisotrop.U[2][2]
+ _atom_site.aniso_U[2][3]
  ⇌ _atom_site_anisotrop.U[2][3]
+ _atom_site.aniso_U[3][3]
  ⇌ _atom_site_anisotrop.U[3][3]
  _atom_site.attached_hydrogens
  _atom_site.auth_asym_id
  _atom_site.auth_atom_id
  _atom_site.auth_comp_id
  _atom_site.auth_seq_id
+ _atom_site.B_equiv_geom_mean
+ _atom_site.B_iso_or_equiv
  _atom_site.calc_attached_atom
  _atom_site.calc_flag
+ _atom_site.Cartn_x
+ _atom_site.Cartn_y
+ _atom_site.Cartn_z
  _atom_site.chemical_conn_number
  → _chemical_conn_atom.number
  _atom_site.constraints
  _atom_site.details (∼ _atom_site_description)
  _atom_site.disorder_assembly
  _atom_site.disorder_group
  _atom_site.footnote_id
+ _atom_site.fract_x
+ _atom_site.fract_y
+ _atom_site.fract_z
  _atom_site.group_PDB
  _atom_site.label_alt_id
  → _atom_sites_alt.id
  _atom_site.label_asym_id
  → _struct_asym.id
  _atom_site.label_atom_id
  → _chem_comp_atom.atom_id
  _atom_site.label_comp_id
  → _chem_comp.id
  _atom_site.label_entity_id
  → _entity.id
  _atom_site.label_seq_id
  → _entity_poly_seq.num
+ _atom_site.occupancy
  _atom_site.refinement_flags
  _atom_site.refinement_flags_adp
  _atom_site.refinement_flags_occupancy
  _atom_site.refinement_flags_posn
  _atom_site.restraints
  _atom_site.symmetry_multiplicity
  _atom_site.thermal_displace_type
  _atom_site.type_symbol
  → _atom_type.symbol
+ _atom_site.U_equiv_geom_mean
+ _atom_site.U_iso_or_equiv
  _atom_site.Wyckoff_symbol

(*b*) ATOM_SITE_ANISOTROP

• _atom_site_anisotrop.id
+ _atom_site_anisotrop.B[1][1] (∼ _atom_site_aniso_B_11)
+ _atom_site_anisotrop.B[1][2] (∼ _atom_site_aniso_B_12)
+ _atom_site_anisotrop.B[1][3] (∼ _atom_site_aniso_B_13)
+ _atom_site_anisotrop.B[2][2] (∼ _atom_site_aniso_B_22)
+ _atom_site_anisotrop.B[2][3] (∼ _atom_site_aniso_B_23)
+ _atom_site_anisotrop.B[3][3] (∼ _atom_site_aniso_B_33)
  _atom_site_anisotrop.ratio (∼ _atom_site_aniso_ratio)
  → _atom_site.id
  _atom_site_anisotrop.type_symbol
  (∼ _atom_site_aniso_type_symbol)
  → _atom_type.symbol
+ _atom_site_anisotrop.U[1][1] (∼ _atom_site_aniso_U_11)
+ _atom_site_anisotrop.U[1][2] (∼ _atom_site_aniso_U_12)
+ _atom_site_anisotrop.U[1][3] (∼ _atom_site_aniso_U_13)
+ _atom_site_anisotrop.U[2][2] (∼ _atom_site_aniso_U_22)
+ _atom_site_anisotrop.U[2][3] (∼ _atom_site_aniso_U_23)
+ _atom_site_anisotrop.U[3][3] (∼ _atom_site_aniso_U_33)

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ∼ symbol. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string _esd to the data name listed. The double arrow (⇌) indicates alternative names in a distinct category.

The refined coordinates of the atoms in the crystallographic asymmetric unit are stored in the ATOM_SITE category. Atom positions and their associated uncertainties may be given using either Cartesian or fractional coordinates, and anisotropic displacement factors and occupancies may be given for each position.

The relationships between categories describing atom sites are shown in Fig. 3.6.7.1.

Several of the mmCIF data names arise from the need to associate atom sites with residues and chains. As in the core CIF dictionary, the identifier for the atom site is the data item **_atom_site_label**. To accommodate standard practice in macromolecular crystallography, the mmCIF atom identifier is the aggregate of **_atom_site.label_alt_id**, ***.label_asym_id**, ***.label_atom_id**, ***.label_comp_id** and ***.label_seq_id**. For the two types of files to be compatible, the data item **_atom_site.id**, which is independent of the different modes of identifying atoms (discussed below), was introduced. The mmCIF identifier **_atom_site.id** is aliased to the core CIF identifier **_atom_site_label**.

Since the identifier does not need to be a number, it is quite possible (although it is not recommended) to use a complex label with an internal structure corresponding to the label components that the mmCIF dictionary provides as separate data items. This scheme is described in Section 3.2.4.1.1. However, normal practice in mmCIFs should be to label sites with the functional components available and to assign a simple numeric sequence to the values of **_atom_site.id** (see Example 3.6.7.1).

In addition to labelling information, each entry in the ATOM_SITE list must contain a value for the data item **_atom_site.type_symbol**, which is a pointer to the table of element symbols in the ATOM_TYPE category. All other data items in the ATOM_SITE category are optional, but it is normal practice to

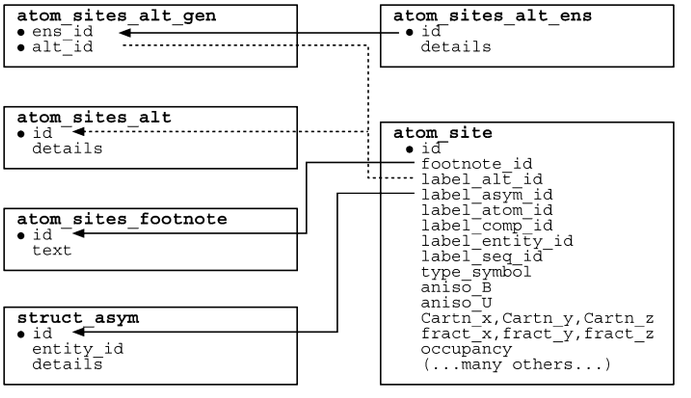


Fig. 3.6.7.1. The family of categories used to describe atom sites. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

Example 3.6.7.1. *Part of the coordinate list for an HIV-1 protease structure (PDB 5HVP) described with data items in the* ATOM_SITE *category. Atoms are given for both polymer and non-polymer regions of the structure, and atoms in the side chain of residue 12 adopt alternative conformations.*

```
loop_
_atom_site.group_PDB
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_seq_id
_atom_site.label_alt_id
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.footnote_id
_atom_site.auth_seq_id
_atom_site.id
 ATOM N N   THR  A  12  .  26.095  32.930  14.590
      1.00  18.97  4  12 8
 ATOM C CA  THR  A  12  .  25.734  32.995  16.032
      1.00  19.80  4  12 9
 ATOM C C   THR  A  12  .  24.695  34.106  16.113
      1.00  20.92  4  12 10
 ATOM O O   THR  A  12  .  24.869  35.118  15.421
      1.00  21.84  4  12 11
 ATOM C CB  THR  A  12  .  26.911  33.346  17.018
      1.00  20.51  4  12 12
 ATOM O OG1 THR  A  12  3  27.946  33.921  16.183
      0.50  20.29  4  12 13
 ATOM O OG1 THR  A  12  4  27.769  32.142  17.103
      0.50  20.59  4  12 14
 ATOM C CG2 THR  A  12  3  27.418  32.181  17.878
      0.50  20.47  4  12 15
 ATOM C CG2 THR  A  12  4  26.489  33.778  18.426
      0.50  20.00  4  12 16
# - - - abbreviated - - -
 HETATM C C1 APS  C  .  1  4.171  29.012   7.116
      0.58  17.27  1 300 101
 HETATM C C2 APS  C  .  1  4.949  27.758   6.793
      0.58  16.95  1 300 102
 HETATM O O3 APS  C  .  1  4.800  26.678   7.393
      0.58  16.85  1 300 103
 HETATM N N4 APS  C  .  1  5.930  27.841   5.869
      0.58  16.43  1 300 104
# - - - abbreviated - - -
```

give either the Cartesian or fractional coordinates. Most macromolecular structures use Cartesian coordinates. Isotropic displacement factors are normally placed directly in the ATOM_SITE category, using **_atom_site.B_iso_or_equiv**. Anisotropic displacement factors may be placed directly in the ATOM_SITE category *or* in the ATOM_SITE_ANISOTROP category. *U*'s may be used instead of *B*'s. It is not acceptable to use both *U*'s and *B*'s, nor is it acceptable to have anisotropic displacement factors in both the ATOM_SITE category and the ATOM_SITE_ANISOTROP category.

Each atom within each chemical component is uniquely identified using the data item **_atom_site.label_atom_id**, which is a reference to the data item **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

The specific object in the asymmetric unit to which the atom belongs is indicated using the data item **_atom_site.label_asym_id**, which is a reference to the data item **_struct_asym.id** in the STRUCT_ASYM category. For macromolecules, it is useful to think of this identifier as a chain ID.

The chemical component to which the atom belongs is indicated using the data item **_atom_site.label_comp_id**, which is a reference to the data item **_chem_comp.id** in the CHEM_COMP category. The chemical component that is referenced in this way may be either a non-polymer or a monomer in a polymer; if it is a monomer in a polymer, it is useful to think of this identifier as the residue name.

The correspondence between the sequence of an entity in a polymer and the sequence information in the coordinate list (and in the STRUCT categories) is established using the data item **_atom_site.label_seq_id**, which is a reference to the data item **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category. This identifier has no meaning for entities that are not part of a polymer; in a polymer it is useful to think of this identifier as the residue number. Note that this is strictly a number. If the combination of a number with an insertion code is needed, **_atom_site.auth_seq_id** should be used (see below).

An alternative set of identifiers can be used for the ***_asym_id**, ***_atom_id**, ***_comp_id** and ***_seq_id** identifiers, but not for ***_alt_id**. The **_atom_site.label_*** data names are standard; there are rules for these identifiers such as the requirement that residue numbers are sequential integers. Different databases may also have their own rules. However, the author of an mmCIF may wish to use a nonstandard labelling scheme, *e.g.* to reflect the residue numbering scheme of a structure to which the present structure is homologous, apart from insertions and gaps. Another situation in which a nonstandard labelling scheme might be used is to follow a local convention for atom names in a non-polymer, such as a haem, that conflicts with the scheme required by a database in which the structure is to be deposited. In these situations, alternative identifiers can be given using the data names (**_atom_site.auth_***).

In regions of the structure with alternative conformations, the specific conformation to which an atom belongs can be indicated using the data item **_atom_site.label_alt_id**, which is a reference to the data item **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

The chemically distinct part of the structure (*e.g.* polymer chain, ligand, solvent) to which an atom belongs can be indicated using the data item **_atom_site.label_entity_id**, which is a reference to the data item **_entity.id** in the ENTITY category.

Most of the information that needs to be associated with an atom site is conveyed by the values of specific data names in mmCIF. However, for historical reasons, a pointer to additional free-text information about an atom site or about a group of atom sites can be given using the data item **_atom_site.footnote_id**, which is a reference to the data item **_atom_sites_footnote.id** in the ATOM_SITES_FOOTNOTE category.

The data item **_atom_site.group_PDB** is a place holder for the tags used by the PDB to identify types of coordinate records. It allows interconversion between mmCIFs and PDB format files. The only permitted values are ATOM and HETATM.

As in the core CIF dictionary, anisotropic displacement parameters in an mmCIF can be given in the same list as the atom positions and occupancies, or can be given in a separate list. However, DDL2 does not permit the same data names to be used for both constructs. Therefore, in mmCIF, anisotropic displacement parameters presented in a separate list are handled in a separate category with its own key, **_atom_site_anisotrop.id**, which must match a corresponding label in the atom-site list, **_atom_site.id**.

The individual elements of the anisotropic displacement matrix are labelled slightly differently in the mmCIF dictionary than in the core CIF dictionary in order to emphasize their matrix character. However, the definitions of the corresponding data items are identical in the two dictionaries.

3.6.7.1.2. *Collections of atom sites*

The data items in these categories are as follows:

(*a*) ATOM_SITES

- **_atom_sites.entry_id**
  → **_entry.id**
  _atom_sites.Cartn_transf_matrix[1][1]
  (∼ **_atom_sites_Cartn_tran_matrix_11**)
  _atom_sites.Cartn_transf_matrix[1][2]
  (∼ **_atom_sites_Cartn_tran_matrix_12**)
  _atom_sites.Cartn_transf_matrix[1][3]
  (∼ **_atom_sites_Cartn_tran_matrix_13**)
  _atom_sites.Cartn_transf_matrix[2][1]
  (∼ **_atom_sites_Cartn_tran_matrix_21**)
  _atom_sites.Cartn_transf_matrix[2][2]
  (∼ **_atom_sites_Cartn_tran_matrix_22**)
  _atom_sites.Cartn_transf_matrix[2][3]
  (∼ **_atom_sites_Cartn_tran_matrix_23**)
  _atom_sites.Cartn_transf_matrix[3][1]
  (∼ **_atom_sites_Cartn_tran_matrix_31**)
  _atom_sites.Cartn_transf_matrix[3][2]
  (∼ **_atom_sites_Cartn_tran_matrix_32**)
  _atom_sites.Cartn_transf_matrix[3][3]
  (∼ **_atom_sites_Cartn_tran_matrix_33**)
  _atom_sites.Cartn_transf_vector[1]
  (∼ **_atom_sites_Cartn_tran_vector_1**)
  _atom_sites.Cartn_transf_vector[2]
  (∼ **_atom_sites_Cartn_tran_vector_2**)
  _atom_sites.Cartn_transf_vector[3]
  (∼ **_atom_sites_Cartn_tran_vector_3**)
  _atom_sites.Cartn_transform_axes
  _atom_sites.fract_transf_matrix[1][1]
  (∼ **_atom_sites_fract_tran_matrix_11**)
  _atom_sites.fract_transf_matrix[1][2]
  (∼ **_atom_sites_fract_tran_matrix_12**)
  _atom_sites.fract_transf_matrix[1][3]
  (∼ **_atom_sites_fract_tran_matrix_13**)
  _atom_sites.fract_transf_matrix[2][1]
  (∼ **_atom_sites_fract_tran_matrix_21**)
  _atom_sites.fract_transf_matrix[2][2]
  (∼ **_atom_sites_fract_tran_matrix_22**)
  _atom_sites.fract_transf_matrix[2][3]
  (∼ **_atom_sites_fract_tran_matrix_23**)
  _atom_sites.fract_transf_matrix[3][1]
  (∼ **_atom_sites_fract_tran_matrix_31**)
  _atom_sites.fract_transf_matrix[3][2]
  (∼ **_atom_sites_fract_tran_matrix_32**)
  _atom_sites.fract_transf_matrix[3][3]
  (∼ **_atom_sites_fract_tran_matrix_33**)
  _atom_sites.fract_transf_vector[1]
  (∼ **_atom_sites_fract_tran_vector_1**)
  _atom_sites.fract_transf_vector[2]
  (∼ **_atom_sites_fract_tran_vector_2**)
  _atom_sites.fract_transf_vector[3]
  (∼ **_atom_sites_fract_tran_vector_3**)
  _atom_sites.solution_hydrogens
  _atom_sites.solution_primary
  _atom_sites.solution_secondary
  _atom_sites.special_details

(*b*) ATOM_SITES_FOOTNOTE

- **_atom_sites_footnote.id**
  _atom_sites_footnote.text

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ∼ symbol.

The ATOM_SITES category of the core dictionary, which is used to record information that applies collectively to all the atom sites in the model of the structure, is incorporated without change into the mmCIF dictionary, and Section 3.2.4.1.2 can be consulted for details.

In practice, the data names in the PHASING categories are preferred to the aliases to the core CIF data items **_atom_sites.solution_primary**, ***_secondary** and ***_hydrogens**. The data items in the mmCIF PHASING categories are designed to allow a much more detailed description of how a macromolecular structure was solved.

Example 3.6.7.2. *Footnotes for particular groups of atom sites in an HIV-1 protease structure (PDB 5HVP) using data items in the ATOM_SITES_FOOTNOTE category.*

```
loop_
  _atom_sites_footnote.id
  _atom_sites_footnote.text
     3
; The positions of these water molecules correlate
  with the alternative orientations of the
  inhibitor. Water molecules with alternative ID
  "1" and occupancy 0.58 correlate with
  inhibitor orientation "1". Water molecules with
  alternative ID "2" and occupancy 0.42 correlate
  with inhibitor orientation "2".
;
     4
; Side chains of these residues adopt alternative
  orientations that do not correlate with the
  alternative orientation of the inhibitor.
;
```

The data item **_atom_sites.entry_id** has been added to the ATOM_SITES category to provide the formal category key required by the DDL2 data model.

The ATOM_SITES_FOOTNOTE category can be used to note something about a group of sites in the ATOM_SITE coordinate list, each of which is flagged with the same value of **_atom_site.footnote_id**. For example, an author may wish to note atoms for which the electron density is very weak, or atoms for which static disorder has been modelled. Example 3.6.7.2 shows how an author has used these data items to describe alternative orientations in part of a structure. However, the very large number of data names describing specific structural characteristics in the mmCIF dictionary mean that these rather general data names are rarely needed.

3.6.7.1.3. *Atom types*

The data items in this category are as follows:

ATOM_TYPE

- **_atom_type.symbol**
  _atom_type.analytical_mass_percent
  (∼ **_atom_type_analytical_mass_%**)
  _atom_type.description
  _atom_type.number_in_cell
  _atom_type.oxidation_number
  _atom_type.radius_bond
  _atom_type.radius_contact
  _atom_type.scat_Cromer_Mann_a1
  _atom_type.scat_Cromer_Mann_a2
  _atom_type.scat_Cromer_Mann_a3
  _atom_type.scat_Cromer_Mann_a4
  _atom_type.scat_Cromer_Mann_b1
  _atom_type.scat_Cromer_Mann_b2
  _atom_type.scat_Cromer_Mann_b3
  _atom_type.scat_Cromer_Mann_b4
  _atom_type.scat_Cromer_Mann_c
  _atom_type.scat_dispersion_imag
  _atom_type.scat_dispersion_real
  _atom_type.scat_dispersion_source
  _atom_type.scat_length_neutron
  _atom_type.scat_source
  _atom_type.scat_versus_stol_list

The bullet (•) indicates a category key. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ∼ symbol.

The ATOM_TYPE category, which provides information about the atomic species associated with each atom site in the model of the structure, is used in the same way in the mmCIF dictionary as in the core CIF dictionary. See Section 3.2.4.1.3 for details.

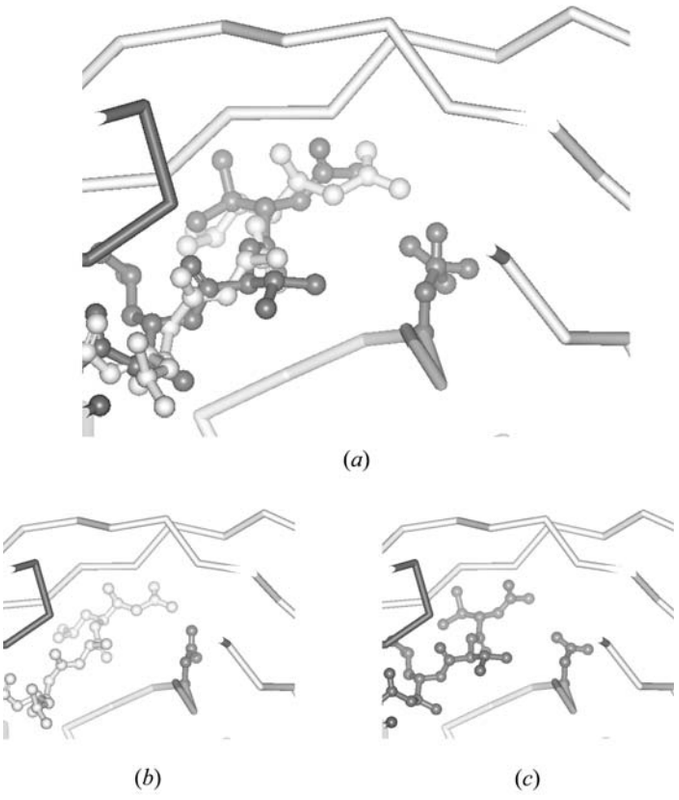


Fig. 3.6.7.2. Alternative conformations in an HIV-1 protease structure (PDB 5HVP) to be described with data items in the ATOM_SITES_ALT, ATOM_SITES_ALT_ENS and ATOM_SITES_ALT_GEN categories. (*a*) Complete structure, (*b*) ensemble 1, (*c*) ensemble 2.

3.6.7.1.4. *Alternative conformations*

The data items in these categories are as follows:

(*a*) ATOM_SITES_ALT

- **_atom_sites_alt.id**
  _atom_sites_alt.details

(*b*) ATOM_SITES_ALT_ENS

- **_atom_sites_alt_ens.id**
  _atom_sites_alt_ens.details

(*c*) ATOM_SITES_ALT_GEN

- **_atom_sites_alt_gen.alt_id**
  → **_atom_sites_alt.id**
- **_atom_sites_alt_gen.ens_id**
  → **_atom_sites_alt_gen.ens_id**

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

Biological macromolecules are often very flexible, and as the resolution of a structure determination increases, it becomes increasingly possible to model reliably the alternative conformations that the structure adopts. Typically, partial occupancies are assigned to atom sites within the alternative conformations to indicate the relative frequency of occurrence of each conformation. It can, however, be difficult to deduce the possible different conformations of the whole structure from inspection of the atom-site occupancies alone. For instance, a segment of protein main chain might adopt one of three slightly different conformations, and within each conformation a particular side chain might adopt one of two possible conformations, one of which sterically distorts an adjacent residue sequence, while the other does not. The data model in the mmCIF dictionary allows these kinds of correlations in positions to be described.

The relationships between the categories used to describe alternative conformations are shown in Fig. 3.6.7.1.

In the core CIF dictionary, alternative conformations are indicated by using the **_atom_site.disorder_assembly** and ***.disorder_group** data items. Aliases to these data items are present in the mmCIF dictionary, but it is not intended that they should be used to describe disorder in a macromolecular structure.

The model for describing alternative conformations in mmCIF uses the ATOM_SITES_ALT family of categories. Ensembles of correlated alternative conformations can be identified using the category ATOM_SITES_ALT_ENS. Each ensemble is generated from one or more of the alternative conformations given in the list of alternative sites in the ATOM_SITES_ALT category. Data items in the

Example 3.6.7.3. *Alternative conformations in an HIV-1 protease structure (PDB 5HVP) described with data items in the ATOM_SITES_ALT, ATOM_SITES_ALT_ENS and ATOM_SITES_ALT_GEN categories.*

```
loop_
_atom_sites_alt.id
_atom_sites_alt.details
  .
; Atom sites with the alternative ID set to null are
  not modelled in alternative conformations
;
  1
; Atom sites with the alternative ID set to 1 have
  been modelled in alternative conformations with
  respect to atom sites marked with alternative
  ID 2. The conformations of amino-acid side chains
  with alternative ID set to 1 correlate with the
  conformation of the inhibitor marked with
  alternative ID 1. Atoms in these side chains have
  been given an occupancy of 0.58 to match the
  occupancy assigned to the inhibitor.
;
  2
; Atom sites with the alternative ID set to 2 have
  been modelled in alternative conformations with
  respect to atom sites marked with alternative
  ID 1. The conformations of amino-acid side chains
  with alternative ID set to 2 correlate with the
  conformation of the inhibitor marked with
  alternative ID 2. Atoms in these side chains have
  been given an occupancy of 0.42 to match the
  occupancy assigned to the inhibitor.
;

loop_
_atom_sites_alt_ens.id
_atom_sites_alt_ens.details
  'Ensemble 1'
; The inhibitor binds to the enzyme in two, roughly
  twofold symmetric, alternative conformations.

  This conformational ensemble includes the more-
  populated conformation of the inhibitor (ID=1) and
  the amino-acid side chains that correlate with this
  inhibitor conformation.
;
  'Ensemble 2'
; The inhibitor binds to the enzyme in two, roughly
  twofold symmetric, alternative conformations.

  This conformational ensemble includes the less-
  populated conformation of the inhibitor (ID=2) and
  the amino-acid side chains that correlate with this
  inhibitor conformation.
;

loop_
_atom_sites_alt_gen.ens_id
_atom_sites_alt_gen.alt_id
   'Ensemble 1'  .
   'Ensemble 1'  1
   'Ensemble 2'  .
   'Ensemble 2'  2
```

ATOM_SITES_ALT_GEN category explicitly tie together the alternative conformations that contribute to each ensemble. Finally, the atoms in each alternative conformation are identified in the ATOM_SITE category by the data item **_atom_site.label_alt_id**.

The current version of the mmCIF dictionary cannot be used to describe an NMR structure determination completely. However, an mmCIF can be used to store the multiple models usually used to describe a structure determined by NMR using the data items in these categories.

Example 3.6.7.3 is a simplified version of the example given in the mmCIF dictionary (see Fig. 3.6.7.2).

3.6.7.2. Molecular chemistry

The categories describing molecular chemistry are as follows:

Molecular chemistry in the core CIF dictionary (§3.6.7.2.1)

CHEMICAL group
- CHEMICAL
- CHEMICAL_CONN_ATOM
- CHEMICAL_CONN_BOND
- CHEMICAL_FORMULA

Chemical components (§3.6.7.2.2)

CHEM_COMP group
- CHEM_COMP
- CHEM_COMP_ANGLE
- CHEM_COMP_ATOM
- CHEM_COMP_BOND
- CHEM_COMP_CHIR
- CHEM_COMP_CHIR_ATOM
- CHEM_COMP_PLANE
- CHEM_COMP_PLANE_ATOM
- CHEM_COMP_TOR
- CHEM_COMP_TOR_VALUE

Chemical links (§3.6.7.2.3)

CHEM_LINK group
- CHEM_COMP_LINK
- CHEM_LINK
- CHEM_LINK_ANGLE
- CHEM_LINK_BOND
- CHEM_LINK_CHIR
- CHEM_LINK_CHIR_ATOM
- CHEM_LINK_PLANE
- CHEM_LINK_PLANE_ATOM
- CHEM_LINK_TOR
- CHEM_LINK_TOR_VALUE
- ENTITY_LINK

The detailed chemistry of the components of a macromolecular structure can be described using data items in the CHEM_COMP and CHEM_LINK category groups. These mmCIF categories are used in preference to those in the CHEMICAL category group in the core CIF dictionary, as macromolecules are in most cases linked assemblies of a limited number of monomers and so they are most efficiently described by defining the monomers and the links between them, rather than by a formal definition of every bond and angle.

All the categories relevant to molecular chemistry are listed in the summary above; note in particular the presence of the category ENTITY_LINK within the formal CHEM_LINK category group.

3.6.7.2.1. *Molecular chemistry in the core CIF dictionary*

The data items in these categories are as follows:

(*a*) CHEMICAL

- **_chemical.entry_id**
  → **_entry.id**
  _chemical.absolute_configuration
  _chemical.compound_source
  _chemical.melting_point
  _chemical.melting_point_gt
  _chemical.melting_point_lt
  _chemical.name_common
  _chemical.name_mineral
  _chemical.name_structure_type
  _chemical.name_systematic
  _chemical.optical_rotation
  _chemical.properties_biological
  _chemical.properties_physical
+ ***_chemical.temperature_decomposition***
  _chemical.temperature_decomposition_gt
  _chemical.temperature_decomposition_lt
+ ***_chemical.temperature_sublimation***
  _chemical.temperature_sublimation_gt
  _chemical.temperature_sublimation_lt

(*b*) CHEMICAL_CONN_ATOM

- ***_chemical_conn_atom.number***
  _chemical_conn_atom.charge
  _chemical_conn_atom.display_x
  _chemical_conn_atom.display_y
  _chemical_conn_atom.NCA
  _chemical_conn_atom.NH
  _chemical_conn_atom.type_symbol

(*c*) CHEMICAL_CONN_BOND

- ***_chemical_conn_bond.atom_1***
- ***_chemical_conn_bond.atom_2***
  _chemical_conn_bond.type

(*d*) CHEMICAL_FORMULA

- **_chemical_formula.entry_id**
  → **_entry.id**
  _chemical_formula.analytical
  _chemical_formula.iupac
  _chemical_formula.moiety
  _chemical_formula.structural
  _chemical_formula.sum
  _chemical_formula.weight
  _chemical_formula.weight_meas

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_). Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string ***_esd*** *to the data name listed.*

Descriptions of molecular chemistry in an mmCIF are normally made using data items in the CHEM_COMP and CHEM_LINK category groups. The CHEMICAL category group is retained in the mmCIF dictionary solely for consistency with the core CIF dictionary and Section 3.2.4.2 may be consulted for details.

Two of the categories in this group, CHEMICAL_CONN_ATOM and CHEMICAL_CONN_BOND, have existing category keys in the core dictionary. The formal keys **_chemical.entry_id** and **_chemical_formula.entry_id** have been added to CHEMICAL and CHEMICAL_FORMULA, respectively, to provide the category keys required by the DDL2 data model.

It is emphasized that these items will not appear in the description of a macromolecular structure, but they are retained to allow the representation of small-molecule or inorganic structures in the DDL2 formalism of mmCIF.

3.6.7.2.2. *Chemical components*

Data items in these categories are as follows:

(*a*) CHEM_COMP

- **_chem_comp.id**
  _chem_comp.formula
  _chem_comp.formula_weight
  _chem_comp.model_details
  _chem_comp.model_erf

```
  _chem_comp.model_source
  _chem_comp.mon_nstd_class
  _chem_comp.mon_nstd_details
  _chem_comp.mon_nstd_flag
  _chem_comp.mon_nstd_parent
  _chem_comp.mon_nstd_parent_comp_id
        → _chem_comp.id
  _chem_comp.name
  _chem_comp.number_atoms_all
  _chem_comp.number_atoms_nh
  _chem_comp.one_letter_code
  _chem_comp.three_letter_code
  _chem_comp.type
```

(*b*) CHEM_COMP_ANGLE

```
• _chem_comp_angle.atom_id_1
        → _chem_comp_atom.atom_id
• _chem_comp_angle.atom_id_2
        → _chem_comp_atom.atom_id
• _chem_comp_angle.atom_id_3
        → _chem_comp_atom.atom_id
• _chem_comp_angle.comp_id
        → _chem_comp.id
+ _chem_comp_angle.value_angle
+ _chem_comp_angle.value_dist
```

(*c*) CHEM_COMP_ATOM

```
• _chem_comp_atom.atom_id
• _chem_comp_atom.comp_id
        → _chem_comp.id
  _chem_comp_atom.alt_atom_id
  _chem_comp_atom.charge
+ _chem_comp_atom.model_Cartn_x
+ _chem_comp_atom.model_Cartn_y
+ _chem_comp_atom.model_Cartn_z
  _chem_comp_atom.partial_charge
  _chem_comp_atom.substruct_code
  _chem_comp_atom.type_symbol
        → _atom_type.symbol
```

(*d*) CHEM_COMP_BOND

```
• _chem_comp_bond.atom_id_1
        → _chem_comp_atom.atom_id
• _chem_comp_bond.atom_id_2
        → _chem_comp_atom.atom_id
• _chem_comp_bond.comp_id
        → _chem_comp.id
  _chem_comp_bond.value_order
+ _chem_comp_bond.value_dist
```

(*e*) CHEM_COMP_CHIR

```
• _chem_comp_chir.id
• _chem_comp_chir.comp_id
  _chem_comp_chir.atom_id
        → _chem_comp_atom.atom_id
  _chem_comp_chir.atom_config
        → _chem_comp.id
  _chem_comp_chir.number_atoms_all
  _chem_comp_chir.number_atoms_nh
  _chem_comp_chir.volume_flag
+ _chem_comp_chir.volume_three
```

(*f*) CHEM_COMP_CHIR_ATOM

```
• _chem_comp_chir_atom.atom_id
        → _chem_comp_atom.atom_id
• _chem_comp_chir_atom.chir_id
        → _chem_comp_chir.id
• _chem_comp_chir_atom.comp_id
        → _chem_comp.id
  _chem_comp_chir_atom.dev
```

(*g*) CHEM_COMP_LINK

```
• _chem_comp_link.link_id
        → _chem_link.id
  _chem_comp_link.details
  _chem_comp_link.type_comp_1
        → _chem_comp.type
  _chem_comp_link.type_comp_2
        → _chem_comp.type
```

(*h*) CHEM_COMP_PLANE

```
• _chem_comp_plane.id
• _chem_comp_plane.comp_id
        → _chem_comp.id
  _chem_comp_plane.number_atoms_all
  _chem_comp_plane.number_atoms_nh
```

(*i*) CHEM_COMP_PLANE_ATOM

```
• _chem_comp_plane_atom.atom_id
        → _chem_comp_atom.atom_id
• _chem_comp_plane_atom.comp_id
        → _chem_comp.id
• _chem_comp_plane_atom.plane_id
        → _chem_comp_plane.id
+ _chem_comp_plane_atom.dist
```

(*j*) CHEM_COMP_TOR

```
• _chem_comp_tor.id
• _chem_comp_tor.comp_id
        → _chem_comp.id
  _chem_comp_tor.atom_id_1
        → _chem_comp_atom.atom_id
  _chem_comp_tor.atom_id_2
        → _chem_comp_atom.atom_id
  _chem_comp_tor.atom_id_3
        → _chem_comp_atom.atom_id
  _chem_comp_tor.atom_id_4
        → _chem_comp_atom.atom_id
```

(*k*) CHEM_COMP_TOR_VALUE

```
• _chem_comp_tor_value.comp_id
• _chem_comp_tor_value.tor_id
+ _chem_comp_tor_value.angle
        → _chem_comp_atom.comp_id
+ _chem_comp_tor_value.dist
        → _chem_comp_tor.id
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string **_esd** *to the data name listed.*

Data items in the CHEM_COMP and related categories allow the covalent geometry, stereochemistry and Cartesian coordinates for the chemical components of the structure to be specified. These components may be monomers, *e.g.* the amino acids that form proteins, the nucleotides that form nucleic acids or the sugars that form oligosaccharides, or they may be the small-molecule compounds, ions or water molecules that co-crystallize with the macromolecule(s).

In a small-molecule structure determination, the chemistry is often deduced from the electron density distribution. In contrast, in macromolecular crystallography, the chemistry of the monomers that form a polymeric macromolecule is usually known in advance and is used to interpret the electron density. In many cases, the chemistry of the monomers is so well determined that it is not worth storing a copy of the geometric restraints used in every mmCIF that uses the same set of data for the monomers. In these cases, the data item **_chem_comp.model_erf** can be used to identify an external reference file (e.r.f.) that contains standard chemical data for these monomers. Although the present version of the mmCIF dictionary does not specify the form that the file identifier might take, it is likely that users will specify the location of the file in their local file system or the URL of files of reference data accessible over the Internet. In the long term, it would be helpful to have a standard repository of reference data for monomers with a stable identifier that is independent of file names or access protocols.

The relationships between the categories used to describe chemical components are shown in Fig. 3.6.7.3.

The CHEM_COMP category provides data items for the chemical formula and formula weight of each component, the total number

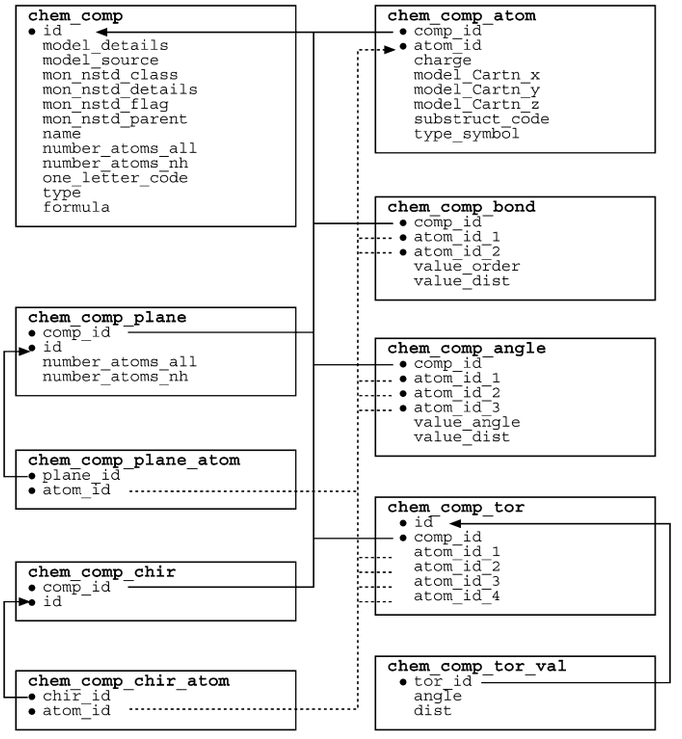


Fig. 3.6.7.3. The family of categories used to describe the chemical and structural features of the monomers and small molecules used to build a model of a structure. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

of atoms, the number of non-hydrogen atoms, and the name of the component. The name of the component will typically be a common name such as 'alanine' or 'valine'; it is recommended that the IUPAC name is used for components that are not among the usual monomers that make up proteins, nucleic acids or sugars.

The one-letter or three-letter code for a standard component may be given (using **_chem_comp.one_letter_code** and **_chem_comp.three_letter_code**, respectively). Values of X for the one-letter code or UNK for the three-letter code are used to indicate components that do not have a standard abbreviation. A component that has been formed by modification of a standard component can be indicated by prefixing the code with a plus sign. A value of '.', which means 'not applicable', should be used for components that are not monomers from which a polymeric macromolecule is built, for example co-crystallized small molecules, ions or water.

The data item **_chem_comp.type** can be used to describe the structural role of a monomer within a polymeric molecule. The types that are recognized are classified as linking monomers (for proteins, nucleic acids and sugars), monomers with an N-terminal or C-terminal cap (for proteins), and monomers with a 5′ or 3′ terminal cap (for nucleic acids). The specification of types for sugars is less complete than for proteins and nucleic acids and no types of terminal groups are currently specified for sugars. The values non-polymer and other are provided for types that have not been defined explicitly.

Information about the source of the model for the chemical component can be given using **_chem_comp.model_source** and **_chem_comp.model_details**. **_chem_comp.model_source** is a text field where the user might, for example, supply a reference to the Cambridge Structural Database or another small-molecule crystallographic database, or describe a molecular-modelling process. **_chem_comp.model_details** can be used to discuss any modification made to the model given in **_chem_comp.model_source**. As mentioned previously, **_chem_comp.model_erf** can be used to specify the location of an external reference file if the model is not described within the current data block.

Macromolecules often contain modifications of standard monomers, such as phosphorylated serines and threonines. In the mmCIF data model, a nonstandard monomer should be treated as a separate CHEM_COMP entry and described in full. However, it may be useful to refer to the standard monomer from which it was derived using the **_chem_comp.mon_nstd_*** data items. There are no fixed rules for what constitutes a 'standard' or 'nonstandard' monomer in this context, but any covalent modification of a standard amino acid or nucleotide would generally be considered nonstandard. Sometimes it is is difficult to decide whether a monomer is standard or nonstandard: selenomethionine is not one of the standard 20 amino acids, but it is so commonly used that geometric restraints for it are included in many standard packages for protein structure refinement.

Data items in the CHEM_COMP_ATOM category can be used to describe the atoms in a component. The position of each atom is given in orthogonal ångström coordinates. These coordinates correspond to the atom positions in the model of the component used in the refinement, not to the final set of refined atom positions recorded in the ATOM_SITE list.

Other CHEM_COMP_ATOM data items can be used to specify what element the atom is and its formal electronic charge, or partial charge. A code may also be assigned to the atom to indicate its role within a substructural classification of the component. The allowed codes are main and side for the main-chain and side-chain parts of amino acids, and base, phos and sugar for the base, phosphate and sugar parts of nucleotides. Atoms that do not belong to a substructure may be assigned the code none.

Data items in the CHEM_COMP_BOND category can be used to describe the intramolecular bonds between atoms in a component. Bond restraints may be described by the distance between the bonded atoms, the bond order, or both. The recognized bond types are the same as those for the core CIF dictionary data item **_chemical_conn_bond.type**, and they fulfil the same role: to characterize a model that could be used for database substructure searching, rather than to give a detailed description of unusual bond types.

In the CHEM_COMP_ANGLE category, atom 2 defines the vertex of the angle involving atoms 1, 2 and 3. The angle may be described as either an angle at the vertex atom or as a distance between atoms 1 and 3.

Data items in the CHEM_COMP_CHIR category can be used to describe the conformation of chiral centres within the component. The absolute configuration and the chiral volume may be specified, as well as the total number of atoms and the number of non-hydrogen atoms bonded to the chiral centre. There is also a flag to indicate whether a restrained chiral volume should match the target value in sign as well as in magnitude. Because chiral centres can involve a variable number of atoms, a separate list of the atoms should be given in CHEM_COMP_CHIR_ATOM.

Data items in the CHEM_COMP_PLANE category can be used to define planes within a component. The number of non-hydrogen atoms and the total number of atoms in each plane can be recorded. The atoms defining each plane should be listed separately in CHEM_COMP_PLANE_ATOM.

Data items in the CHEM_COMP_TOR category can be used to give details about the torsion angles in a component. A torsion angle may be described either as an angle or as a distance between the first and last atoms. (A torsion angle cannot be completely described by a distance, but sometimes a distance

Example 3.6.7.4. *The description of a component (adriamycin) of a macromolecule with data items in the* CHEM_COMP, CHEM_COMP_ATOM, CHEM_COMP_BOND, CHEM_COMP_TOR *and* CHEM_COMP_TOR_VALUE *categories (Leonard et al., 1993).*

```
_chem_comp.id                 'DM2'
_chem_comp.name               'adriamycin'
_chem_comp.type                non-polymer
_chem_comp.formula            'C27 H29 N1 O11'
_chem_comp.number_atoms_all  68
_chem_comp.number_atoms_nh   39
_chem_comp.formula_weight    543.51

loop_
_chem_comp_atom.comp_id
_chem_comp_atom.atom_id
_chem_comp_atom.type_symbol
_chem_comp_atom.model_Cartn_x
_chem_comp_atom.model_Cartn_y
_chem_comp_atom.model_Cartn_z
   DM2 'C1'  C  12.996  0.476  12.694
   DM2 'C2'  C  13.982 -0.225  13.183
   DM2 'C3'  C  12.482  0.165  11.515
# - - - abbreviated - - -

loop_
_chem_comp_bond.comp_id
_chem_comp_bond.atom_id_1
_chem_comp_bond.atom_id_2
_chem_comp_bond.value_order
_chem_comp_bond.value_dist
_chem_comp_bond.value_dist_esd
   DM2 'C1' 'C2' sing  1.517  0.0210
   DM2 'C2' 'C3' sing  1.445  0.0040
# - - - abbreviated - - -

loop_
  _chem_comp_tor.comp_id
  _chem_comp_tor.id
  _chem_comp_tor.atom_id_1
  _chem_comp_tor.atom_id_2
  _chem_comp_tor.atom_id_3
  _chem_comp_tor.atom_id_4
    phe  phe_chi1   N    CA   CB   CG
    phe  phe_chi2   CA   CB   CG   CD1
    phe  phe_ring1  CB   CG   CD1  CE1
    phe  phe_ring2  CB   CG   CD2  CE2
    phe  phe_ring3  CG   CD1  CE1  CZ
    phe  phe_ring4  CD1  CE1  CZ   CE2
    phe  phe_ring5  CE1  CZ   CE2  CD2

loop_
  _chem_comp_tor_value.tor_id
  _chem_comp_tor_value.comp_id
  _chem_comp_tor_value.angle
  _chem_comp_tor_value.dist
    phe_chi1   phe  -60.0  2.88
    phe_chi1   phe  180.0  3.72
    phe_chi1   phe   60.0  2.88
    phe_chi2   phe   90.0  3.34
    phe_chi2   phe  -90.0  3.34
    phe_ring1  phe  180.0  3.75
    phe_ring2  phe  180.0  3.75
    phe_ring3  phe    0.0  2.80
    phe_ring4  phe    0.0  2.80
    phe_ring5  phe    0.0  2.80
```

restraint is used in refinement, where the value of the angle is assumed to be close to the target value.) As torsion angles can have more than one target value, the target values are specified in the CHEM_COMP_TOR_VALUE category.

Data items in the CHEM_COMP_LINK category can be used to provide a table of links between the components of the structure. Each link is assigned an identifier (`_chem_comp_link.link_id`) and the types of monomer at each end of the link are stated. The types are those allowed for the parent data item `_chem_comp.type`.

The use of many of these data items to describe a typical component is shown in Example 3.6.7.4.

3.6.7.2.3. *Chemical links*

The data items in these categories are as follows:

(*a*) CHEM_LINK

```
• _chem_link.id
  _chem_link.details
```

(*b*) CHEM_LINK_ANGLE

```
• _chem_link_angle.atom_id_1
• _chem_link_angle.atom_id_2
• _chem_link_angle.atom_id_3
• _chem_link_angle.link_id
        → _chem_link.id
  _chem_link_angle.atom_1_comp_id
  _chem_link_angle.atom_2_comp_id
  _chem_link_angle.atom_3_comp_id
+ _chem_link_angle.value_angle
+ _chem_link_angle.value_dist
```

(*c*) CHEM_LINK_BOND

```
• _chem_link_bond.atom_id_1
• _chem_link_bond.atom_id_2
• _chem_link_bond.link_id
        → _chem_link.id
  _chem_link_bond.atom_1_comp_id
  _chem_link_bond.atom_2_comp_id
+ _chem_link_bond.value_dist
  _chem_link_bond.value_order
```

(*d*) CHEM_LINK_CHIR

```
• _chem_link_chir.id
• _chem_link_chir.link_id
        → _chem_link.id
  _chem_link_chir.atom_comp_id
  _chem_link_chir.atom_id
  _chem_link_chir.atom_config
  _chem_link_chir.number_atoms_all
  _chem_link_chir.number_atoms_nh
  _chem_link_chir.volume_flag
+ _chem_link_chir.volume_three
```

(*e*) CHEM_LINK_CHIR_ATOM

```
• _chem_link_chir_atom.atom_id
• _chem_link_chir_atom.chir_id
        → _chem_link_chir.id
  _chem_link_chir_atom.atom_comp_id
  _chem_link_chir_atom.dev
```

(*f*) CHEM_LINK_PLANE

```
• _chem_link_plane.id
• _chem_link_plane.link_id
        → _chem_link.id
  _chem_link_plane.number_atoms_all
  _chem_link_plane.number_atoms_nh
```

(*g*) CHEM_LINK_PLANE_ATOM

```
• _chem_link_plane_atom.atom_id
• _chem_link_plane_atom.plane_id
        → _chem_link_plane.id
  _chem_link_plane_atom.atom_comp_id
```

(*h*) CHEM_LINK_TOR

```
• _chem_link_tor.id
• _chem_link_tor.link_id
        → _chem_link.id
  _chem_link_tor.atom_1_comp_id
  _chem_link_tor.atom_2_comp_id
  _chem_link_tor.atom_3_comp_id
  _chem_link_tor.atom_4_comp_id
  _chem_link_tor.atom_id_1
  _chem_link_tor.atom_id_2
  _chem_link_tor.atom_id_3
  _chem_link_tor.atom_id_4
```

(*i*) CHEM_LINK_TOR_VALUE

```
• _chem_link_tor_value.tor_id
        → _chem_link_tor.id
+ _chem_link_tor_value.angle
+ _chem_link_tor_value.dist
```

(*j*) ENTITY_LINK

```
• _entity_link.link_id
        → _chem_link.id
  _entity_link.details
  _entity_link.entity_id_1
        → _entity.id
  _entity_link.entity_id_2
        → _entity.id
  _entity_link.entity_seq_num_1
        → _entity_poly_seq.num
  _entity_link.entity_seq_num_2
        → _entity_poly_seq.num
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. Data items marked with a plus (+) have companion data names for the standard uncertainty in the reported value, formed by appending the string ***_esd*** *to the data name listed.*

The geometry of the links between chemical components or entities can be described in the CHEM_LINK group of categories. Chemical components may be linked together according to the type of the component; defining the linking according to the type of the component rather than by each component in turn allows a type of polymer link for all the monomers in a polymer to be specified (*e.g.* L-peptide linking). The geometry of the links can be specified in the remaining CHEM_LINK categories. The relationships between categories used to describe links between chemical components are shown in Fig. 3.6.7.4, which also shows how information about the links is passed to the CHEM_COMP and CHEM_LINK categories. For simplicity, the categories CHEM_COMP_PLANE, CHEM_COMP_PLANE_ATOM, CHEM_COMP_CHIR, CHEM_COMP_CHIR_ATOM and ENTITY_LINK are not included in Fig. 3.6.7.4.

Note that this category group can be used to describe the links that connect the monomers within a macromolecular polymer (using the CHEM_LINK categories) and also the intramolecular links between separate molecules in the whole complex (using the ENTITY_LINK category). Intramolecular links, for example a covalent bond formed between a bound ligand and an amino-acid side chain, are usually discovered as a result of the structure determination, and it would therefore seem more appropriate to describe them in the STRUCT_CONN category. However, since one of the roles of the CHEM_LINK category group is to record target values used for restraints or constraints during the refinement of the model of the structure, ideal values for the geometry of any entity-to-entity links should be given here.

Data items in the CHEM_LINK category are used to assign a unique identifier to each link and allow the author to record any unusual aspects of each link. The other categories in the CHEM_LINK category group describe the geometric model of each link, and are closely analogous to the similarly named categories in the CHEM_COMP group.

The relationships among these categories are complex (see Fig. 3.6.7.4). Each atom that participates in an aspect of the link (for example, a bond, an angle, a chiral centre, a torsion angle or a plane) must be identified and it must also be specified whether the atom is in the first or second of the components that form the link.

Data items in the CHEM_LINK_BOND category describe the bonds between atoms participating in an intermolecular link between chemical components. Bond restraints may be described by the distance between the bonded atoms, the bond order or both.

An angle at a link may be described in the CHEM_LINK_ANGLE category as either an angle at the vertex atom or as a distance between the atoms attached to the vertex. For data items in both the CHEM_LINK_BOND and CHEM_LINK_ANGLE categories, a target

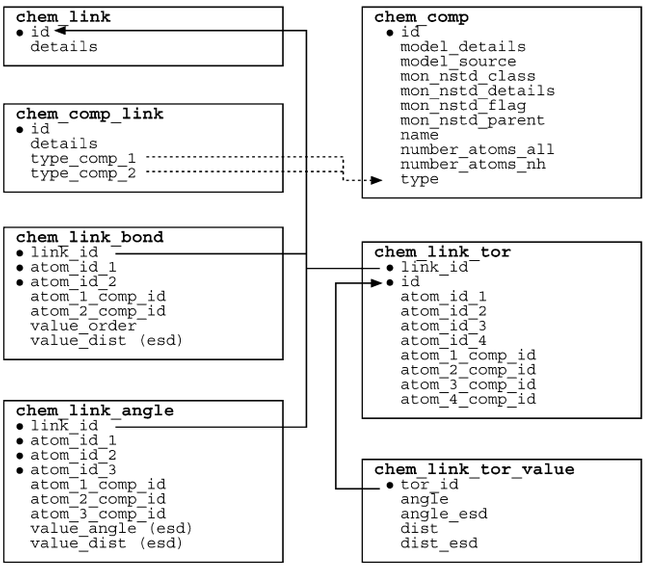


Fig. 3.6.7.4. The family of categories used to describe the links between chemical components. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

value and its associated standard uncertainty may be specified (Example 3.6.7.5).

Data items in the CHEM_LINK_CHIR category can be used to describe the conformation of chiral centres in a link between two chemical components. The absolute configuration and the chiral volume may be specified, as well as the total number of atoms and the number of non-hydrogen atoms bonded to the chiral centre. There is also a flag to indicate whether a restrained chiral volume should match the target value in sign as well as in magnitude. Because chiral centres can involve a variable number of atoms, a separate list of the atoms should be given in CHEM_LINK_CHIR_ATOM.

Data items in the CHEM_LINK_PLANE category can be used to list planes defined across a link between two chemical components. Because planes can involve a variable number of atoms, a separate list of the atoms should be given in CHEM_LINK_PLANE_ATOM.

Data items in the CHEM_LINK_TOR category can be used to give details of the torsion angles across a link between two chemical

Example 3.6.7.5. *A peptide bond described with data items in the CHEM_LINK_BOND and CHEM_LINK_ATOM categories.*

```
loop_
  _chem_link_bond.link_id
  _chem_link_bond.value_dist
  _chem_link_bond.value_dist_esd
  _chem_link_bond.atom_id_1
  _chem_link_bond.atom_1_comp_id
  _chem_link_bond.atom_id_2
  _chem_link_bond.atom_2_comp_id
    PEPTIDE  1.329  0.014  C  1  N  2

loop_
  _chem_link_angle.link_id
  _chem_link_angle.value_angle
  _chem_link_angle.value_angle_esd
  _chem_link_angle.atom_id_1
  _chem_link_angle.atom_1_comp_id
  _chem_link_angle.atom_id_2
  _chem_link_angle.atom_2_comp_id
  _chem_link_angle.atom_id_3
  _chem_link_angle.atom_3_comp_id
    PEPTIDE  116.2  2.0  CA 1  C  1  N  2
    PEPTIDE  123.0  1.6  O  1  C  1  N  2
    PEPTIDE  121.7  1.8  C  1  N  2  CA 2
```

components. The torsion angle may be described either as an angle or as a distance between the first and last atoms. As torsion angles can have more than one target value, the target values are specified in the CHEM_LINK_TOR_VALUE category.

The ENTITY_LINK category is used to identify the participants in links between distinct molecular entities. A pointer to the details of the link is given in **_entity_link.link_id**, which matches a value of **_chem_link.id** in the CHEM_LINK category.

3.6.7.3. Distinct chemical species

The categories describing distinct chemical entities are as follows:

ENTITY group
Entities (§3.6.7.3.1)
ENTITY
ENTITY_KEYWORDS
ENTITY_NAME_COM
ENTITY_NAME_SYS
ENTITY_SRC_GEN
ENTITY_SRC_NAT
Polymer entities (§3.6.7.3.2)
ENTITY_POLY
ENTITY_POLY_SEQ

The ENTITY categories of the mmCIF dictionary should be used in preference to the CHEMICAL categories of the core CIF dictionary. In a typical small-molecule structure determination, for which the core CIF dictionary was designed, the substance being studied can be thought of as a single chemical species, even if it contains distinct ions or ligands. In a macromolecular structure, it is more often the case that separate descriptions are appropriate for each of the distinct chemical species that comprise the structural complex. The ENTITY categories allow the species present and their basic chemical properties to be specified. Their structures and connectivity are described in other categories.

It is important, therefore, to remember that the ENTITY data do not represent the result of the crystallographic experiment; those results are given using the ATOM_SITE data items and are discussed and described using data items in the STRUCT family of categories. The ENTITY categories describe the chemistry of the molecules under investigation and are most usefully considered as the ideal groups to which the structure is restrained or constrained during refinement.

It is also important to remember that entities do not correspond directly to the total contents of the asymmetric unit. Entities are described only once, even in structures in which the entity occurs several times. The STRUCT_ASYM data items, which reference the list of entities, describe and label the contents of the asymmetric unit.

The following discussion treats the data items used for entities in general (Section 3.6.7.3.1) and those used more specifically to describe polymeric entities (Section 3.6.7.3.2) separately.

3.6.7.3.1. *Description of entities*

The data items in these categories are as follows:

(*a*) ENTITY
- **_entity.id**

_entity.details
_entity.formula_weight
_entity.src_method
_entity.type

(*b*) ENTITY_KEYWORDS
- **_entity_keywords.entity_id**
  → **_entity.id**
- **_entity_keywords.text**

(*c*) ENTITY_NAME_COM
- **_entity_name_com.entity_id**
  → **_entity.id**
- **_entity_name_com.name**

(*d*) ENTITY_NAME_SYS
- **_entity_name_sys.entity_id**
  → **_entity.id**
- **_entity_name_sys.name**

_entity_name_sys.system

(*e*) ENTITY_SRC_GEN
- **_entity_src_gen.entity_id**
  → **_entity.id**

_entity_src_gen.gene_src_common_name
_entity_src_gen.gene_src_details
_entity_src_gen.gene_src_genus
_entity_src_gen.gene_src_species
_entity_src_gen.gene_src_strain
_entity_src_gen.gene_src_tissue
_entity_src_gen.gene_src_tissue_fraction
_entity_src_gen.host_org_common_name
_entity_src_gen.host_org_details
_entity_src_gen.host_org_genus
_entity_src_gen.host_org_species
_entity_src_gen.host_org_strain
_entity_src_gen.plasmid_details
_entity_src_gen.plasmid_name

(*f*) ENTITY_SRC_NAT
- **_entity_src_nat.entity_id**
  → **_entity.id**

_entity_src_nat.common_name
_entity_src_nat.details
_entity_src_nat.genus
_entity_src_nat.species
_entity_src_nat.strain
_entity_src_nat.tissue
_entity_src_nat.tissue_fraction

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

An entity in mmCIF is a chemically distinct molecular component of the structural complex described in the mmCIF. The three possible types of molecular entities are polymer, non-polymer and water. Note that the 'water' entity is water, and only water. Any other well ordered solvent molecules or ions should be treated as non-polymer entities. The relationships between categories used to describe the features of entities are shown in Fig. 3.6.7.5, which also shows how the information describing the entity is linked to the coordinate list in the ATOM_SITE category.

Data items in the ENTITY category are used to label each distinct chemical molecule with a reference code (**_entity.id**), to give the formula weight in daltons (if available) and to define the type of the entity as one of polymer, non-polymer or water. The method by which the entity was produced may be indicated using the item **_entity.src_method**, whose allowed values are `nat` (indicating that the sample was isolated from a natural source), `man` (indicating a genetically manipulated source) or `syn` (indicating a chemical synthesis). A value of `nat` indicates that additional details should be given in the ENTITY_SRC_NAT category and a value of `man` indicates that additional details should be given in the ENTITY_SRC_GEN category. As these flags are only relevant to the macromolecular entities of a structural complex, a value of '.', indicating 'inapplicable', should be given to **_entity.src_method** for solvent or water molecules. The **_entity.details** field can be used for a free-text description of any special features of the entity.

Keywords characterizing the individual molecular species may be given using data items in the ENTITY_KEYWORD category. These keywords should only be used to record information that does not depend on knowledge of the molecular structure. Thus a polypeptide could be described as a polypeptide, or an enzyme, or

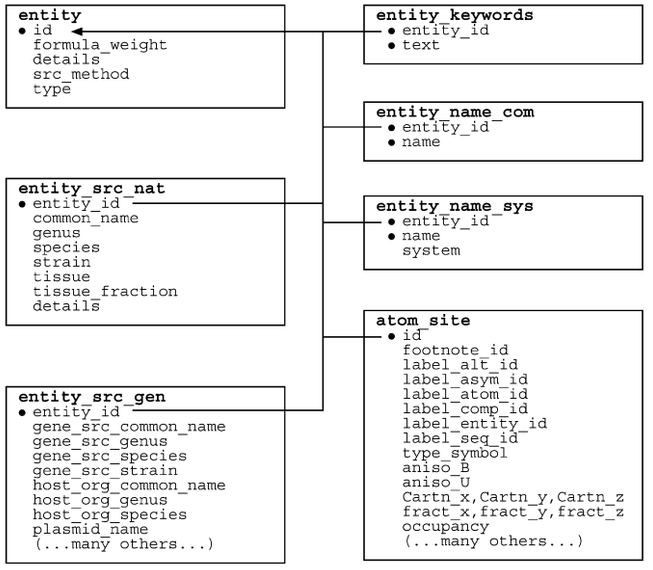


Fig. 3.6.7.5. The family of categories used to describe chemical entities. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data item.

a protease, but it should not be described as an $\alpha\beta$-barrel; a number of categories within the STRUCT family allow keywords specific to the structure of the macromolecule to be given.

Data items in the ENTITY_NAME_COM category may be used to give any common names for an entity. Several different names can be recorded for each entity if appropriate.

Similarly, data items in the ENTITY_NAME_SYS category may be used to give systematic names for each entity. Again, several different names can be recorded for each entity if appropriate. The data item **_entity_name_sys.system** can be used to record the system according to which the systematic name was generated.

The ENTITY_SRC_GEN category allows a description of the source of entities produced by genetic manipulation to be given. There are data items for describing the tissue from which the gene was obtained, the plasmid into which it was incorporated for expression, and the host organism in which the macromolecule was expressed (Example 3.6.7.6).

The ENTITY_SRC_NAT category allows a description of the source of entities obtained from a natural tissue to be given. Data items are provided for the common and systematic name (by genus, species and, where relevant, strain) of the organism from which the material was obtained. Other data items can be used to describe the tissue (and if necessary the subcellular fraction of the tissue) from which the entity was isolated.

Example 3.6.7.6. *An example of the description of the entities in an HIV-1 protease structure (PDB 5HVP), described using data items in the ENTITY, ENTITY_NAME_COM, ENTITY_NAME_SYS and ENTITY_SRC_GEN categories.*

```
loop_
_entity.id
_entity.type
_entity.formula_weight
_entity.details
   1  polymer       10916
; The enzymatically competent form of HIV protease is
  a dimer. This entity corresponds to one monomer of
  an active dimer.
;
   2  non-polymer   647.2    .
   3  water            18    .

loop_
_entity_name_com.entity_id
_entity_name_com.name
   1  'HIV-1 protease monomer'
   1  'HIV-1 PR monomer'
   2  'acetyl-pepstatin'
   2  'acetyl-Ile-Val-Asp-Statine-Ala-Ile-Statine'
   3  'water'

_entity_name_sys.entity_id       1
_entity_name_sys.name           'EC 2.1.1.1'
_entity_name_sys.system         'Enzyme convention'

loop_
_entity_src_gen.entity_id
_entity_src_gen.gene_src_common_name
_entity_src_gen.gene_src_strain
_entity_src_gen.host_org_common_name
_entity_src_gen.host_org_genus
_entity_src_gen.host_org_species
_entity_src_gen.plasmid_name
1 'HIV-1' 'NY-5' 'bacteria' 'Escherichia' 'coli'
'pB322'
```

3.6.7.3.2. *Polymer entities*

The data items in these categories are as follows:

(*a*) ENTITY_POLY

- **_entity_poly.entity_id**
  → **_entity.id**

_entity_poly.nstd_chirality
_entity_poly.nstd_linkage
_entity_poly.nstd_monomer
_entity_poly.number_of_monomers
_entity_poly.type
_entity_poly.type_details

(*b*) ENTITY_POLY_SEQ

- **_entity_poly_seq.entity_id**
  → **_entity.id**
- **_entity_poly_seq.mon_id**
  → **_chem_comp.id**
- **_entity_poly_seq.num**

_entity_poly_seq.hetero

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

The polymer type, sequence length and information about any nonstandard features of the polymer may be specified using data items in the ENTITY_POLY category. The sequence of monomers in each polymer entity is given using data items in the ENTITY_POLY_SEQ category. The relationships between categories describing polymer entities are shown in Fig. 3.6.7.6, which also shows how the information describing the polymer is linked to the coordinate list in the ATOM_SITE category and to the full chemical description of each monomer or nonstandard monomer in the CHEM_COMP category.

Non-polymer entities are treated as individual chemical components, in the same way in which monomers within a polymer are treated as individual chemical components. They may be fully described in the CHEM_COMP group of categories (Example 3.6.7.7).

Data items in the ENTITY_POLY category can be used to give the number of monomers in the polymer and to assign the type of the polymer as one of the set of types `polypeptide(D)`, `polypeptide(L)`, `polydeoxyribonucleotide`, `polyribonucleotide`, `polysaccharide(D)`, `polysaccharide(L)` or `other`. Details of deviations from a standard type may be given in **_entity_poly.type_details**.

In some cases, the polymer is best described as one of the standard types even if it contains some nonstandard features. Flags are provided to indicate the presence of three types of nonstandard features. The presence of chiral centres other than those implied

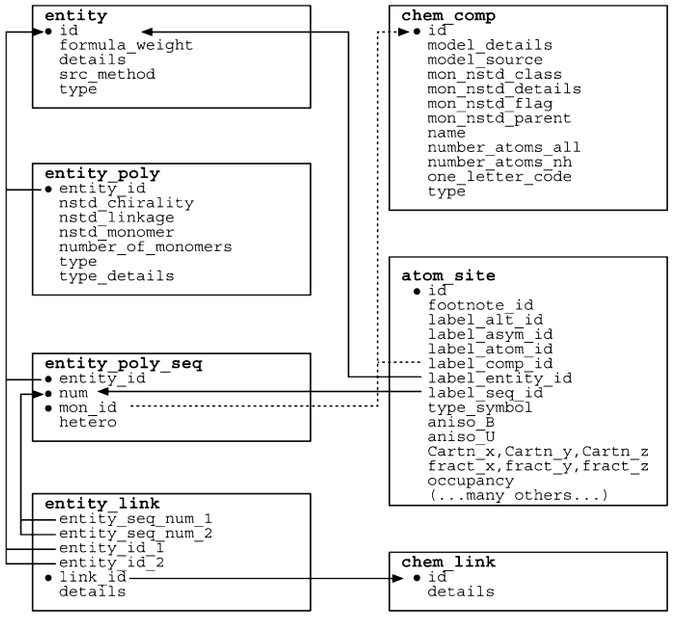


Fig. 3.6.7.6. The family of categories used to describe polymer chemical entities. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

Example 3.6.7.7. *An example of both polymer and non-polymer entities in a drug–DNA complex (NDB DDF040) described with data items in the* ENTITY, ENTITY_KEYWORDS, ENTITY_NAME_COM, ENTITY_POLY *and* ENTITY_POLY_SEQ *categories (Narayana et al., 1991).*

```
loop_
_entity.id
_entity.type
_entity.src_method
   1  polymer     man
   2  non-polymer man
   3  water       .

loop_
_entity_keywords.entity_id
_entity_keywords.text
   1  'nucleic acid'
   2  'drug'

loop_
_entity_name_com.entity_id
_entity_name_com.name
   2  adriamycin
   3  water

loop_
_entity_poly.entity_id
_entity_poly.number_of_monomers
_entity_poly.type
   1  8  'polydeoxyribonucleotide'

loop_
_entity_poly_seq.entity_id
_entity_poly_seq.mon_id
_entity_poly_seq.num
   1  T  1
   1  G  2
   1  G  3
   1  C  4
   1  C  5
   1  A  6
# - - - abbreviated - - -
```

by the assigned type is indicated by assigning a value of `yes` to the data item **`_entity_poly.nstd_chirality`**. A value of `yes` for **`_entity_poly.nstd_linkage`** indicates the presence of monomer-to-monomer links different from those implied by the assigned type and a value of `yes` for **`_entity_poly.nstd_monomer`** indicates the presence of one or more nonstandard monomer components.

Data items in the ENTITY_POLY_SEQ category describe the sequence of monomers in a polymer. By including **`_entity_poly_seq.mon_id`** in the category key, it is possible to allow for sequence heterogeneity by allowing a given sequence number to be correlated with more than one monomer ID. Sequence heterogeneity is shown in the example of crambin in Section 3.6.3.

3.6.7.4. Molecular or packing geometry

The categories describing geometry are as follows:

GEOM group
- GEOM
- GEOM_ANGLE
- GEOM_BOND
- GEOM_CONTACT
- GEOM_HBOND
- GEOM_TORSION

The categories within the GEOM group are used in the core CIF dictionary to describe the geometry of the model that results from the structure determination, and can be used to select values that will be published in a report describing the structure. The complexity of macromolecular structures means that a different approach to presenting the results of a structure determination is needed. The STRUCT family of categories was created to meet this need. The GEOM categories are retained in the mmCIF dictionary, but only for consistency with the core CIF dictionary.

The data items in the categories in the GEOM group are:

(*a*) GEOM

- **`_geom.entry_id`**
  → **`_entry.id`**

  `_geom.details` (∼ ***`_geom_special_details`***)

(*b*) GEOM_ANGLE

- ***`_geom_angle.atom_site_id_1`***
  (∼ ***`_geom_angle_atom_site_label_1`***)
- ***`_geom_angle.atom_site_id_2`***
  (∼ ***`_geom_angle_atom_site_label_2`***)
- ***`_geom_angle.atom_site_id_3`***
  (∼ ***`_geom_angle_atom_site_label_3`***)
- ***`_geom_angle.site_symmetry_1`***
- ***`_geom_angle.site_symmetry_2`***
- ***`_geom_angle.site_symmetry_3`***

`_geom_angle.atom_site_auth_asym_id_1`
→ **`_atom_site.auth_asym_id`**
`_geom_angle.atom_site_auth_atom_id_1`
→ **`_atom_site.auth_atom_id`**
`_geom_angle.atom_site_auth_comp_id_1`
→ **`_atom_site.auth_comp_id`**
`_geom_angle.atom_site_auth_seq_id_1`
→ **`_atom_site.auth_seq_id`**
`_geom_angle.atom_site_auth_asym_id_2`
→ **`_atom_site.auth_asym_id`**
`_geom_angle.atom_site_auth_atom_id_2`
→ **`_atom_site.auth_atom_id`**
`_geom_angle.atom_site_auth_comp_id_2`
→ **`_atom_site.auth_comp_id`**
`_geom_angle.atom_site_auth_seq_id_2`
→ **`_atom_site.auth_seq_id`**
`_geom_angle.atom_site_auth_asym_id_3`
→ **`_atom_site.auth_asym_id`**
`_geom_angle.atom_site_auth_atom_id_3`
→ **`_atom_site.auth_atom_id`**
`_geom_angle.atom_site_auth_comp_id_3`
→ **`_atom_site.auth_comp_id`**
`_geom_angle.atom_site_auth_seq_id_3`
→ **`_atom_site.auth_seq_id`**
→ **`_atom_site.id`**
`_geom_angle.atom_site_label_alt_id_1`
→ **`_atom_site.label_alt_id`**
`_geom_angle.atom_site_label_asym_id_1`
→ **`_atom_site.label_asym_id`**
`_geom_angle.atom_site_label_atom_id_1`
→ **`_atom_site.label_atom_id`**

_geom_angle.atom_site_label_comp_id_1
→ _atom_site.label_comp_id
_geom_angle.atom_site_label_seq_id_1
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_angle.atom_site_label_alt_id_2
→ _atom_site.label_alt_id
_geom_angle.atom_site_label_asym_id_2
→ _atom_site.label_asym_id
_geom_angle.atom_site_label_atom_id_2
→ _atom_site.label_atom_id
_geom_angle.atom_site_label_comp_id_2
→ _atom_site.label_comp_id
_geom_angle.atom_site_label_seq_id_2
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_angle.atom_site_label_alt_id_3
→ _atom_site.label_alt_id
_geom_angle.atom_site_label_asym_id_3
→ _atom_site.label_asym_id
_geom_angle.atom_site_label_atom_id_3
→ _atom_site.label_atom_id
_geom_angle.atom_site_label_comp_id_3
→ _atom_site.label_comp_id
_geom_angle.atom_site_label_seq_id_3
→ _atom_site.label_seq_id
_geom_angle.publ_flag
\+ _geom_angle.value (∼ _geom_angle)

(*c*) GEOM_BOND

• _geom_bond.atom_site_id_1
(∼ _geom_bond_atom_site_label_1)
• _geom_bond.atom_site_id_2
(∼ _geom_bond_atom_site_label_2)
• _geom_bond.site_symmetry_1
• _geom_bond.site_symmetry_2
_geom_bond.atom_site_auth_asym_id_1
→ _atom_site.auth_asym_id
_geom_bond.atom_site_auth_atom_id_1
→ _atom_site.auth_atom_id
_geom_bond.atom_site_auth_comp_id_1
→ _atom_site.auth_comp_id
_geom_bond.atom_site_auth_seq_id_1
→ _atom_site.auth_seq_id
_geom_bond.atom_site_auth_asym_id_2
→ _atom_site.auth_asym_id
_geom_bond.atom_site_auth_atom_id_2
→ _atom_site.auth_atom_id
_geom_bond.atom_site_auth_comp_id_2
→ _atom_site.auth_comp_id
_geom_bond.atom_site_auth_seq_id_2
→ _atom_site.auth_seq_id
→ _atom_site.id
_geom_bond.atom_site_label_alt_id_1
→ _atom_site.label_alt_id
_geom_bond.atom_site_label_asym_id_1
→ _atom_site.label_asym_id
_geom_bond.atom_site_label_atom_id_1
→ _atom_site.label_atom_id
_geom_bond.atom_site_label_comp_id_1
→ _atom_site.label_comp_id
_geom_bond.atom_site_label_seq_id_1
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_bond.atom_site_label_alt_id_2
→ _atom_site.label_alt_id
_geom_bond.atom_site_label_asym_id_2
→ _atom_site.label_asym_id
_geom_bond.atom_site_label_atom_id_2
→ _atom_site.label_atom_id
_geom_bond.atom_site_label_comp_id_2
→ _atom_site.label_comp_id
_geom_bond.atom_site_label_seq_id_2
→ _atom_site.label_seq_id
\+ _geom_bond.dist (∼ _geom_bond_distance)
_geom_bond.publ_flag
_geom_bond.valence

(*d*) GEOM_CONTACT

• _geom_contact.atom_site_id_1
(∼ _geom_contact_atom_site_label_1)
• _geom_contact.atom_site_id_2
(∼ _geom_contact_atom_site_label_2)
• _geom_contact.site_symmetry_1
• _geom_contact.site_symmetry_2
_geom_contact.atom_site_auth_asym_id_1
→ _atom_site.auth_asym_id
_geom_contact.atom_site_auth_atom_id_1
→ _atom_site.auth_atom_id
_geom_contact.atom_site_auth_comp_id_1
→ _atom_site.auth_comp_id
_geom_contact.atom_site_auth_seq_id_1
→ _atom_site.auth_seq_id
_geom_contact.atom_site_auth_asym_id_2
→ _atom_site.auth_asym_id
_geom_contact.atom_site_auth_atom_id_2
→ _atom_site.auth_atom_id
_geom_contact.atom_site_auth_comp_id_2
→ _atom_site.auth_comp_id
_geom_contact.atom_site_auth_seq_id_2
→ _atom_site.auth_seq_id
→ _atom_site.id
_geom_contact.atom_site_label_alt_id_1
→ _atom_site.label_alt_id
_geom_contact.atom_site_label_asym_id_1
→ _atom_site.label_asym_id
_geom_contact.atom_site_label_atom_id_1
→ _atom_site.label_atom_id
_geom_contact.atom_site_label_comp_id_1
→ _atom_site.label_comp_id
_geom_contact.atom_site_label_seq_id_1
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_contact.atom_site_label_alt_id_2
→ _atom_site.label_alt_id
_geom_contact.atom_site_label_asym_id_2
→ _atom_site.label_asym_id
_geom_contact.atom_site_label_atom_id_2
→ _atom_site.label_atom_id
_geom_contact.atom_site_label_comp_id_2
→ _atom_site.label_comp_id
_geom_contact.atom_site_label_seq_id_2
→ _atom_site.label_seq_id
\+ _geom_contact.dist (∼ _geom_contact_distance)
_geom_contact.publ_flag

(*e*) GEOM_HBOND

• _geom_hbond.atom_site_id_A
→ _atom_site.id
• _geom_hbond.atom_site_id_D
→ _atom_site.id
• _geom_hbond.atom_site_id_H
→ _atom_site.id
• _geom_hbond.site_symmetry_A
• _geom_hbond.site_symmetry_D
• _geom_hbond.site_symmetry_H
\+ _geom_hbond.angle_DHA
_geom_hbond.atom_site_auth_asym_id_A
→ _atom_site.auth_asym_id
_geom_hbond.atom_site_auth_atom_id_A
→ _atom_site.auth_atom_id
_geom_hbond.atom_site_auth_comp_id_A
→ _atom_site.auth_comp_id
_geom_hbond.atom_site_auth_seq_id_A
→ _atom_site.auth_seq_id
_geom_hbond.atom_site_auth_asym_id_D
→ _atom_site.auth_asym_id
_geom_hbond.atom_site_auth_atom_id_D
→ _atom_site.auth_atom_id
_geom_hbond.atom_site_auth_comp_id_D
→ _atom_site.auth_comp_id
_geom_hbond.atom_site_auth_seq_id_D
→ _atom_site.auth_seq_id
_geom_hbond.atom_site_auth_asym_id_H
→ _atom_site.auth_asym_id
_geom_hbond.atom_site_auth_atom_id_H
→ _atom_site.auth_atom_id
_geom_hbond.atom_site_auth_comp_id_H
→ _atom_site.auth_comp_id
_geom_hbond.atom_site_auth_seq_id_H
→ _atom_site.auth_seq_id
_geom_hbond.atom_site_label_alt_id_A
→ _atom_site.label_alt_id
_geom_hbond.atom_site_label_asym_id_A
→ _atom_site.label_asym_id

_geom_hbond.atom_site_label_atom_id_A
→ _atom_site.label_atom_id
_geom_hbond.atom_site_label_comp_id_A
→ _atom_site.label_comp_id
_geom_hbond.atom_site_label_seq_id_A
→ _atom_site.label_seq_id
_geom_hbond.atom_site_label_alt_id_D
→ _atom_site.label_alt_id
_geom_hbond.atom_site_label_asym_id_D
→ _atom_site.label_asym_id
_geom_hbond.atom_site_label_atom_id_D
→ _atom_site.label_atom_id
_geom_hbond.atom_site_label_comp_id_D
→ _atom_site.label_comp_id
_geom_hbond.atom_site_label_seq_id_D
→ _atom_site.label_seq_id
_geom_hbond.atom_site_label_alt_id_H
→ _atom_site.label_alt_id
_geom_hbond.atom_site_label_asym_id_H
→ _atom_site.label_asym_id
_geom_hbond.atom_site_label_atom_id_H
→ _atom_site.label_atom_id
_geom_hbond.atom_site_label_comp_id_H
→ _atom_site.label_comp_id
_geom_hbond.atom_site_label_seq_id_H
→ _atom_site.label_seq_id
\+ _geom_hbond.dist_DA (∼ _geom_hbond_distance_DA)
\+ _geom_hbond.dist_DH (∼ _geom_hbond_distance_DH)
\+ _geom_hbond.dist_HA (∼ _geom_hbond_distance_HA)
_geom_hbond.publ_flag

(*f*) GEOM_TORSION

• _geom_torsion.atom_site_id_1
(∼ _geom_torsion_atom_site_label_1)
• _geom_torsion.atom_site_id_2
(∼ _geom_torsion_atom_site_label_2)
• _geom_torsion.atom_site_id_3
(∼ _geom_torsion_atom_site_label_3)
• _geom_torsion.atom_site_id_4
(∼ _geom_torsion_atom_site_label_4)
• _geom_torsion.site_symmetry_1
• _geom_torsion.site_symmetry_2
• _geom_torsion.site_symmetry_3
• _geom_torsion.site_symmetry_4
_geom_torsion.atom_site_auth_asym_id_1
→ _atom_site.auth_asym_id
_geom_torsion.atom_site_auth_atom_id_1
→ _atom_site.auth_atom_id
_geom_torsion.atom_site_auth_comp_id_1
→ _atom_site.auth_comp_id
_geom_torsion.atom_site_auth_seq_id_1
→ _atom_site.auth_seq_id
_geom_torsion.atom_site_auth_asym_id_2
→ _atom_site.auth_asym_id
_geom_torsion.atom_site_auth_atom_id_2
→ _atom_site.auth_atom_id
_geom_torsion.atom_site_auth_comp_id_2
→ _atom_site.auth_comp_id
_geom_torsion.atom_site_auth_seq_id_2
→ _atom_site.auth_seq_id
_geom_torsion.atom_site_auth_asym_id_3
→ _atom_site.auth_asym_id
_geom_torsion.atom_site_auth_atom_id_3
→ _atom_site.auth_atom_id
_geom_torsion.atom_site_auth_comp_id_3
→ _atom_site.auth_comp_id
_geom_torsion.atom_site_auth_seq_id_3
→ _atom_site.auth_seq_id
_geom_torsion.atom_site_auth_asym_id_4
→ _atom_site.auth_asym_id
_geom_torsion.atom_site_auth_atom_id_4
→ _atom_site.auth_atom_id
_geom_torsion.atom_site_auth_comp_id_4
→ _atom_site.auth_comp_id
_geom_torsion.atom_site_auth_seq_id_4
→ _atom_site.auth_seq_id
→ _atom_site.id
_geom_torsion.atom_site_label_alt_id_1
→ _atom_site.label_alt_id
_geom_torsion.atom_site_label_asym_id_1
→ _atom_site.label_asym_id
_geom_torsion.atom_site_label_atom_id_1
→ _atom_site.label_atom_id
_geom_torsion.atom_site_label_comp_id_1
→ _atom_site.label_comp_id
_geom_torsion.atom_site_label_seq_id_1
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_torsion.atom_site_label_alt_id_2
→ _atom_site.label_alt_id
_geom_torsion.atom_site_label_asym_id_2
→ _atom_site.label_asym_id
_geom_torsion.atom_site_label_atom_id_2
→ _atom_site.label_atom_id
_geom_torsion.atom_site_label_comp_id_2
→ _atom_site.label_comp_id
_geom_torsion.atom_site_label_seq_id_2
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_torsion.atom_site_label_alt_id_3
→ _atom_site.label_alt_id
_geom_torsion.atom_site_label_asym_id_3
→ _atom_site.label_asym_id
_geom_torsion.atom_site_label_atom_id_3
→ _atom_site.label_atom_id
_geom_torsion.atom_site_label_comp_id_3
→ _atom_site.label_comp_id
_geom_torsion.atom_site_label_seq_id_3
→ _atom_site.label_seq_id
→ _atom_site.id
_geom_torsion.atom_site_label_alt_id_4
→ _atom_site.label_alt_id
_geom_torsion.atom_site_label_asym_id_4
→ _atom_site.label_asym_id
_geom_torsion.atom_site_label_atom_id_4
→ _atom_site.label_atom_id
_geom_torsion.atom_site_label_comp_id_4
→ _atom_site.label_comp_id
_geom_torsion.atom_site_label_seq_id_4
→ _atom_site.label_seq_id
_geom_torsion.publ_flag
\+ _geom_torsion.value (∼ _geom_torsion)

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (.) *to an underscore* (_) *except where indicated by the* ∼ *symbol. Data items marked with a plus* (+) *have companion data names for the standard uncertainty in the reported value, formed by appending the string* _esd *to the data name listed.*

3.6.7.5. Molecular structure

The categories describing molecular structure are as follows:

STRUCT group

Higher-level macromolecular structure (§3.6.7.5.1)
- STRUCT
- STRUCT_ASYM
- STRUCT_BIOL
- STRUCT_BIOL_GEN
- STRUCT_BIOL_KEYWORDS
- STRUCT_BIOL_VIEW

Secondary structure (§3.6.7.5.2)
- STRUCT_CONF
- STRUCT_CONF_TYPE

Structural interactions (§3.6.7.5.3)
- STRUCT_CONN
- STRUCT_CONN_TYPE

Structural features of monomers (§3.6.7.5.4)
- STRUCT_MON_DETAILS
- STRUCT_MON_NUCL
- STRUCT_MON_PROT
- STRUCT_MON_PROT_CIS

Noncrystallographic symmetry (§3.6.7.5.5)
- STRUCT_NCS_DOM
- STRUCT_NCS_DOM_LIM
- STRUCT_NCS_ENS

STRUCT_NCS_ENS_GEN
STRUCT_NCS_OPER

External databases (§3.6.7.5.6)
STRUCT_REF
STRUCT_REF_SEQ
STRUCT_REF_SEQ_DIF

β-sheets (§3.6.7.5.7)
STRUCT_SHEET
STRUCT_SHEET_TOPOLOGY
STRUCT_SHEET_ORDER
STRUCT_SHEET_RANGE
STRUCT_SHEET_HBOND

Molecular sites (§3.6.7.5.8)
STRUCT_SITE_GEN
STRUCT_SITE_KEYWORDS
STRUCT_SITE_VIEW

The results of the determination of a structure can be described in mmCIF using data items in the categories contained in the STRUCT category group. This is a very large group of categories and it has been divided into eight groups of related categories for the discussions that follow: (1) those that describe the structure at the level of biologically relevant assemblies; (2) those that describe the secondary structure of the macromolecules present; (3) those that describe the structural interactions that determine the conformation of the macromolecules; (4) those that describe properties of the structure at the monomer level; (5) those that describe ensembles of identical domains related by noncrystallographic symmetry; (6) those that provide references to related entities in external databases; (7) those that describe the β-sheets present in the structure; and (8) those that provide detailed descriptions of the structure of biologically interesting molecular sites.

3.6.7.5.1. *Higher-level macromolecular structure*

The data items in these categories are as follows:

(*a*) STRUCT
- **_struct.entry_id**
  → **_entry.id**

_struct.title

(*b*) STRUCT_ASYM
- **_struct_asym.id**

_struct_asym.details
_struct_asym.entity_id
  → **_entity.id**

(*c*) STRUCT_BIOL
- **_struct_biol.id**

_struct_biol.details

(*d*) STRUCT_BIOL_GEN
- **_struct_biol_gen.asym_id**
  → **_struct_asym.id**
- **_struct_biol_gen.biol_id**
  → **_struct_biol.id**
- **_struct_biol_gen.symmetry**

_struct_biol_gen.details

(*e*) STRUCT_BIOL_KEYWORDS
- **_struct_biol_keywords.biol_id**
  → **_struct_biol.id**
- **_struct_biol_keywords.text**

(*f*) STRUCT_BIOL_VIEW
- **_struct_biol_view.biol_id**
  → **_struct_biol.id**
- **_struct_biol_view.id**

_struct_biol_view.details
_struct_biol_view.rot_matrix[1][1]
_struct_biol_view.rot_matrix[1][2]
_struct_biol_view.rot_matrix[1][3]
_struct_biol_view.rot_matrix[2][1]
_struct_biol_view.rot_matrix[2][2]
_struct_biol_view.rot_matrix[2][3]
_struct_biol_view.rot_matrix[3][1]
_struct_biol_view.rot_matrix[3][2]
_struct_biol_view.rot_matrix[3][3]

(*g*) STRUCT_KEYWORDS
- **_struct_keywords.entry_id**
  → **_entry.id**
- **_struct_keywords.text**

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

The data items in these categories serve two related but distinct purposes.

The first purpose is to label each of the entities in the asymmetric unit, using data items in the STRUCT_ASYM category. These labels become part of the category key that identifies each coordinate record and they are used extensively throughout the STRUCT family of categories, so care must be taken to select a labelling scheme that is concise and informative.

The second function is descriptive. The categories descending from STRUCT_BIOL allow the author of the mmCIF to identify and annotate the biologically relevant structural units found by the structure determination. What constitutes a biological unit can depend on the context. Take the case of a structure with two polymers related by noncrystallographic symmetry, each of which binds a small-molecule cofactor. If the author wishes to describe the dimer interface, the biological unit could be taken to be the two protein molecules. If the author wishes to highlight the cofactor binding mode, the biological unit could be taken to be one protein molecule and its bound cofactor. In this second case, there could be an additional biological unit of the second protein molecule and its bound cofactor, which may or may not be identical in conformation to the first.

The relationships between categories used to describe higher-level structure are illustrated in Fig. 3.6.7.7.

The STRUCT category serves to link the structure to the overall identifier for the data block, using **_struct.entry_id**, and to supply a title that describes the entire structure. The importance of this title as a succinct description of the structure should not be underestimated, and the author should express concisely but clearly in **_struct.title** the components of interest and the importance of this particular study. It is useful to think of this title as describing

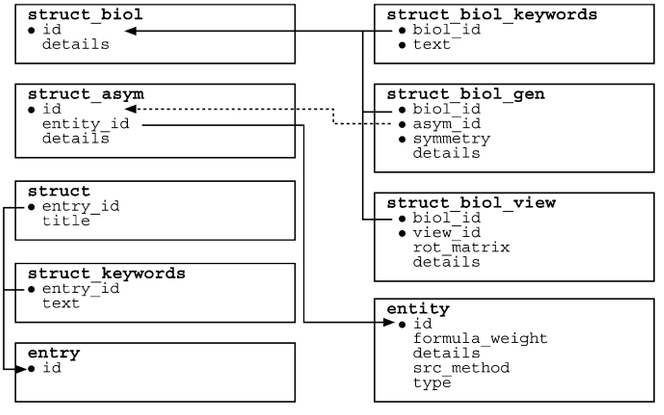


Fig. 3.6.7.7. The family of categories used to describe the higher-level macromolecular structure. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

Example 3.6.7.8. *The higher-level structure of the complex of HIV-1 protease with an inhibitor (PDB 5HVP) described with data items in the* STRUCT_ASYM, STRUCT_BIOL, STRUCT_BIOL_KEYWORDS *and* STRUCT_BIOL_GEN *categories.*

```
loop_
_struct_asym.id
_struct_asym.entity_id
_struct_asym.details
   A  1  'one monomer of the dimeric enzyme'
   B  1  'one monomer of the dimeric enzyme'
   C  2
 'one partially occupied position for the inhibitor'
   D  2
 'one partially occupied position for the inhibitor'

loop_
_struct_biol.id
_struct_biol.details
   1
; significant deviations from twofold symmetry exist
  in this dimeric enzyme
;
   2
; The drug binds to this enzyme in two roughly
  twofold symmetric modes.

  Hence this biological unit (2) is roughly twofold
  symmetric to biological unit (3). Disorder in the
  protein chain indicated with alternative ID 1
  should be used with this biological unit.
;
   3
; The drug binds to this enzyme in two roughly
  twofold symmetric modes.

  Hence this biological unit (3) is roughly twofold
  symmetric to biological unit (2). Disorder in the
  protein chain indicated with alternative ID 2
  should be used with this biological unit.
;

loop_
_struct_biol_gen.biol_id
_struct_biol_gen.asym_id
_struct_biol_gen.symmetry
   1  A  1_555     1  B  1_555
   2  A  1_555     2  B  1_555     2  C  1_555
   3  A  1_555     3  B  1_555     3  D  1_555
```

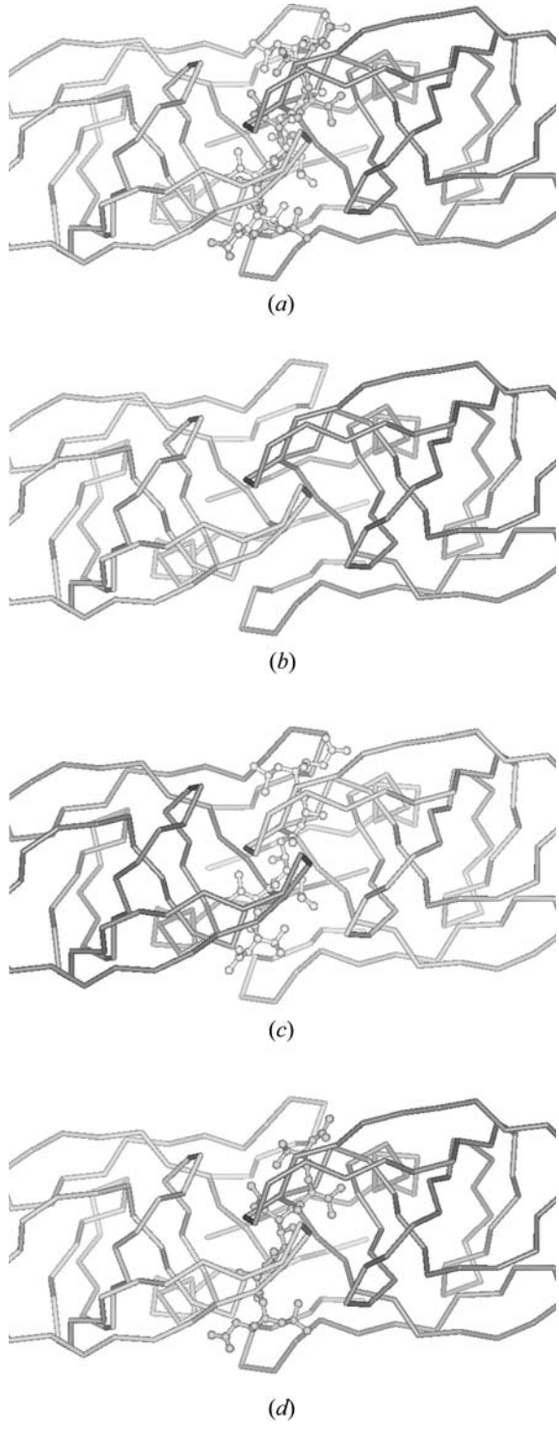


Fig. 3.6.7.8. The higher-level structure of the complex of HIV-1 protease with an inhibitor (PDB 5HVP) to be described with data items in the STRUCT_ASYM, STRUCT_BIOL, STRUCT_BIOL_KEYWORDS and STRUCT_BIOL_GEN categories. (*a*) Complete structure; (*b*), (*c*), (*d*) three different biological units.

the motivation for the structure determination, rather than the result. For instance, if the goal of the study was to determine the structure of enzyme A at pH 7.2 as part of a study of the mechanism of the reaction catalysed by the enzyme, an appropriate value for `_struct.title` would be 'Enzyme A at pH 7.2', even if the structure was found to contain two molecules per asymmetric unit, a bound calcium ion and a disordered loop between residues 47 and 52.

The STRUCT_KEYWORDS category allows an author to include keywords for the structure that has been determined. Other categories, such as STRUCT_BIOL_KEYWORDS and STRUCT_SITE_KEYWORDS, allow more specific keywords to be given, but the STRUCT_KEYWORDS category is the most likely category to be searched by simple information retrieval applications, so the author of an mmCIF might want to duplicate any keywords given elsewhere in the mmCIF in STRUCT_KEYWORDS as well.

The chemical entities that form the contents of the asymmetric unit are identified using data items in the ENTITY categories. The data items in the STRUCT_ASYM category link these entities to the structure itself. A unique identifier is attached to each occurrence of each entity in the asymmetric unit using `_struct_asym.id`. This identifier forms a part of the atom label in the ATOM_SITE category, which is used throughout the many categories in the STRUCT group in describing the structure. The identifier is also used in generating biological assemblies.

The usual reason for determining the structure of a biological macromolecule is to get information about the biologically relevant assemblies of the entities in the crystal structure. These assemblies take many forms and could encompass the complete contents of the asymmetric unit, a fraction of the contents of the asymmetric unit or the contents of more than one asymmetric unit. Each assembly, or 'biological unit', is given an identifier in the STRUCT_BIOL category and the author may annotate each biological unit using the data item `_struct_biol.details`. Key-

words for each biological unit can be given using data items in the STRUCT_BIOL_KEYWORD category.

The entities that comprise the biological unit are specified using data items in the STRUCT_BIOL_GEN category by reference to the appropriate values of **_struct_asym.id** and by specifying any symmetry transformation that must be applied to the entities to generate the biological unit.

Data items in the STRUCT_BIOL_VIEW category allow the author to specify an orientation of the biological unit that provides a useful view of the structure. The comments given in **_struct_biol_view.details** may be used as a figure caption if the view is intended to be a figure in a report describing the structure.

The example of crambin in Section 3.6.3 shows the relations between the categories defining higher-level structure for the straightforward case of a single protein molecule (with a small co-crystallization molecule and solvent) in the asymmetric unit. The structure of HIV-1 protease with a bound inhibitor (PDB 5HVP), shown in Example 3.6.7.8, is considerably more complex. There are two entities: the monomeric form of the enzyme and the small-molecule inhibitor. The asymmetric unit contains two copies of the enzyme monomer (both fully occupied) and two copies of the inhibitor (each of which is partially occupied) (Fig. 3.6.7.8). Three biological assemblies are constructed for this system. One biological unit contains only the dimeric enzyme (Fig. 3.6.7.8*b*), the second contains the dimeric enzyme with one partially occupied conformation of the inhibitor (Fig. 3.6.7.8*c*) and the third contains the dimeric enzyme with the second partially occupied conformation of the inhibitor (Fig. 3.6.7.8*d*). There are alternative conformations of the side chains in the enzyme that correlate with the binding mode of the inhibitor.

3.6.7.5.2. *Secondary structure*

The data items in these categories are as follows:

(*a*) STRUCT_CONF_TYPE

```
• _struct_conf_type.id
  _struct_conf_type.criteria
  _struct_conf_type.reference
```

(*b*) STRUCT_CONF

```
• _struct_conf.id
  _struct_conf.beg_label_asym_id
        → _atom_site.label_asym_id
  _struct_conf.beg_label_comp_id
        → _atom_site.label_comp_id
  _struct_conf.beg_label_seq_id
        → _atom_site.label_seq_id
  _struct_conf.beg_auth_asym_id
        → _atom_site.auth_asym_id
  _struct_conf.beg_auth_comp_id
        → _atom_site.auth_comp_id
  _struct_conf.beg_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_conf.conf_type_id
        → _struct_conf_type.id
  _struct_conf.details
  _struct_conf.end_label_asym_id
        → _atom_site.label_asym_id
  _struct_conf.end_label_comp_id
        → _atom_site.label_comp_id
  _struct_conf.end_label_seq_id
        → _atom_site.label_seq_id
  _struct_conf.end_auth_asym_id
        → _atom_site.auth_asym_id
  _struct_conf.end_auth_comp_id
        → _atom_site.auth_comp_id
  _struct_conf.end_auth_seq_id
        → _atom_site.auth_seq_id
```

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

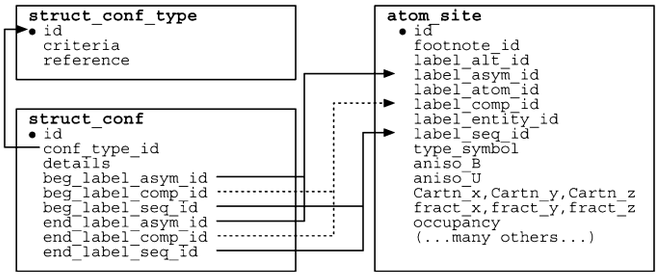


Fig. 3.6.7.9. The family of categories used to describe secondary structure. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

Example 3.6.7.9. *Secondary structure in an HIV-1 protease structure (PDB 5HVP) described with data items in the STRUCT_CONF_TYPE and STRUCT_CONF categories.*

```
loop_
_struct_conf_type.id
_struct_conf_type.criteria
   HELX_RH_AL_P  'author judgement'
   STRN          'author judgement'
   TURN_TY1_P    'author judgement'
   TURN_TY1P_P   'author judgement'
   TURN_TY2_P    'author judgement'
   TURN_TY2P_P   'author judgement'

loop_
_struct_conf.id
_struct_conf.conf_type_id
_struct_conf.beg_label_comp_id
_struct_conf.beg_label_asym_id
_struct_conf.beg_label_seq_id
_struct_conf.end_label_comp_id
_struct_conf.end_label_asym_id
_struct_conf.end_label_seq_id
   HELX1  HELX_RH_AL_P  ARG  A   87  GLN  A   92
   HELX2  HELX_RH_AL_P  ARG  B  287  GLN  B  292
   STRN1  STRN          PRO  A    1  LEU  A    5
   STRN2  STRN          CYS  B  295  PHE  B  299
   STRN3  STRN          CYS  A   95  PHE  A  299
   STRN4  STRN          PRO  B  201  LEU  B  205
   TURN1  TURN_TY1P_P   ILE  A   15  GLN  A   18
   TURN2  TURN_TY2_P    GLY  A   49  GLY  A   52
   TURN3  TURN_TY1P_P   ILE  A   55  HIS  A   69
   TURN4  TURN_TY1_P    THR  A   91  GLY  A   94
```

The primary structure of a macromolecule is defined by the sequence of the components (amino acids, nucleic acids or sugars) in the polymer chain. The polymer chains assume conformations based on the torsion angles adopted by the rotatable bonds in the polymer backbone; the resulting conformations are referred to as the secondary structure of the polymer. Several patterns of values of backbone torsion angles have been described and given names, such as α-helix, β-strand, turn and coil for proteins, and A-, B- and Z-helix for nucleic acids.

In the mmCIF dictionary, these secondary structures are described in the STRUCT_CONF and STRUCT_CONF_TYPE categories. Note that the data items in these categories describe only the secondary structure; the tertiary organization of β-strands into β-sheets is described in the STRUCT_SHEET_* categories. There are no data items for describing the tertiary organization of α-helices or nucleic acids in the current version of the mmCIF dictionary.

The relationships between categories used to describe secondary structure are shown in Fig. 3.6.7.9.

The type of the secondary structure is specified in the STRUCT_CONF_TYPE category, along with the criteria used to identify it. The range of monomers assigned to each secondary-structure element is given in the STRUCT_CONF category.

The allowed values for the data item **_struct_conf_type.id** cover most types of protein and nucleic acid secondary structure (Example 3.6.7.9). The criteria that define the secondary structure may be given using the data item **_struct_conf_type.criteria**. **_struct_conf_type.reference** can be used to specify a reference to the literature in which the criteria are explained in more detail.

The residues that define the beginning and end of each region of secondary structure are identified with the appropriate ***_asym**, ***_comp** and ***_seq** identifiers. The standard labelling system or the author's alternative labelling system may be used. The identification of the residues assigned to each region of secondary structure is linked to the labelling information in the ATOM_SITE category. Unusual features of a conformation may be described using **_struct_conf.details**.

3.6.7.5.3. *Structural interactions*

The data items in these categories are as follows:

(*a*) STRUCT_CONN_TYPE

```
• _struct_conn_type.id
  _struct_conn_type.criteria
  _struct_conn_type.reference
```

(*b*) STRUCT_CONN

```
• _struct_conn.id
  _struct_conn.conn_type_id
        → _struct_conn_type.id
  _struct_conn.details
  _struct_conn.ptnr1_label_alt_id
        → _atom_sites_alt.id
  _struct_conn.ptnr1_label_asym_id
        → _atom_site.label_asym_id
  _struct_conn.ptnr1_label_atom_id
        → _chem_comp_atom.atom_id
  _struct_conn.ptnr1_label_comp_id
        → _atom_site.label_comp_id
  _struct_conn.ptnr1_label_seq_id
        → _atom_site.label_seq_id
  _struct_conn.ptnr1_auth_asym_id
        → _atom_site.auth_asym_id
  _struct_conn.ptnr1_auth_atom_id
        → _atom_site.auth_atom_id
  _struct_conn.ptnr1_auth_comp_id
        → _atom_site.auth_comp_id
  _struct_conn.ptnr1_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_conn.ptnr1_role
  _struct_conn.ptnr1_symmetry
  _struct_conn.ptnr2_label_alt_id
        → _atom_sites_alt.id
  _struct_conn.ptnr2_label_asym_id
        → _atom_site.label_asym_id
  _struct_conn.ptnr2_label_atom_id
        → _chem_comp_atom.atom_id
  _struct_conn.ptnr2_label_comp_id
        → _atom_site.label_comp_id
  _struct_conn.ptnr2_label_seq_id
        → _atom_site.label_seq_id
  _struct_conn.ptnr2_auth_asym_id
        → _atom_site.auth_asym_id
  _struct_conn.ptnr2_auth_atom_id
        → _atom_site.auth_atom_id
  _struct_conn.ptnr2_auth_comp_id
        → _atom_site.auth_comp_id
  _struct_conn.ptnr2_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_conn.ptnr2_role
  _struct_conn.ptnr2_symmetry
```

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

The structural interactions that are described with data items in the STRUCT_CONN family of categories are the tertiary result of a structure determination, not the chemical connectivity of the components of the structure. In general, the interactions described using the STRUCT_CONN data items are noncovalent, such as hydrogen bonds, salt bridges and metal coordination.

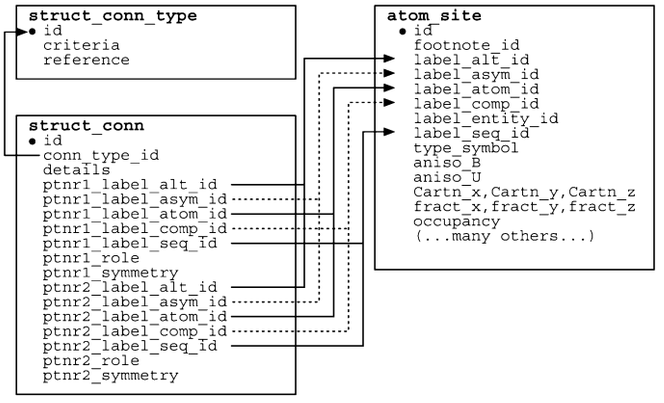


Fig. 3.6.7.10. The family of categories used to describe structural interactions such as hydrogen bonding, salt bridges and disulfide bridges. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

It is useful to think of the structure interactions given in CHEM_COMP_BOND, CHEM_LINK and ENTITY_LINK as the covalent interactions that are known in advance of the structure determination because the chemistry of the components is well defined. Literature or calculated values for these interactions are often used as restraints during the refinement. In contrast, the structural interactions described in the STRUCT_CONN family of categories are not known in advance and are part of the results of the structure determination.

This distinction only holds approximately, as there are clearly bonds, such as disulfide links, that are covalent and usually restrained during the refinement but that are also a result of the folding of the protein revealed by the structure determination, and thus should be described using STRUCT_CONN data items.

In general, the STRUCT_CONN data items would not be used to list all the structure interactions. Instead, the author of the mmCIF would use the STRUCT_CONN data items to identify and annotate only the structural interactions worthy of discussion. The relationships between categories used to describe structural interactions are shown in Fig. 3.6.7.10.

Structural interactions such as hydrogen bonds, salt bridges and disulfide bridges can be described in the STRUCT_CONN category. The type of each interaction and the criteria used to identify the interaction can be specified in the STRUCT_CONN_TYPE category (Example 3.6.7.10).

The atoms participating in each interaction are arbitrarily labelled as 'partner 1' and 'partner 2'. Each is identified by the ***_alt**, ***_asym**, ***_atom**, ***_comp** and ***_seq** constituents of the corresponding atom-site label. The role of each partner in the interaction (*e.g.* donor, acceptor) may be specified, and any crystallographic symmetry operation needed to transform the atom from the position given in the ATOM_SITE list to the position where the interaction occurs can be given. The atoms participating in the interaction may also be identified using an alternative labelling scheme if the author has supplied one.

Unusual aspects of the interaction may be discussed in **_struct_conn.details**. The general type of an interaction can be indicated using **_struct_conn.conn_type_id**, which references one of the standard types described using data items in the STRUCT_CONN_TYPE category.

The specific types of structural connection that may be recorded are those allowed for **_struct_conn_type.id**, namely covalent and hydrogen bonds, ionic (salt-bridge) interactions, disulfide

Example 3.6.7.10. *A hypothetical salt bridge and hydrogen bond described with data items in the* STRUCT_CONN_TYPE *and* STRUCT_CONN *categories.*

```
loop_
_struct_conn_type.id
_struct_conn_type.criteria
   saltbr
; negative to positive distance > 2.5 Angstroms,
  < 3.2 Angstroms
;
   hydrog
; N-O distance > 2.5 Angstroms, < 3.5 Angstroms,
  N-O-C  angle < 120 degrees
;

loop_
_struct_conn.id
_struct_conn.conn_type_id
_struct_conn.ptnr1_label_comp_id
_struct_conn.ptnr1_label_asym_id
_struct_conn.ptnr1_label_seq_id
_struct_conn.ptnr1_label_atom_id
_struct_conn.ptnr1_role
_struct_conn.ptnr1_symmetry
_struct_conn.ptnr2_label_comp_id
_struct_conn.ptnr2_label_asym_id
_struct_conn.ptnr2_label_seq_id
_struct_conn.ptnr2_label_atom_id
_struct_conn.ptnr2_role
_struct_conn.ptnr2_symmetry
  C1 saltbr ARG  A  87 NZ1 positive 1_555
            GLU  A  92 OE1 negative 1_555
  C2 hydrog ARG  B 287 N   donor    1_555
            GLY  B 292 O   acceptor 1_555
```

links, metal coordination, mismatched base pairs, covalent residue modifications and covalent modifications of nucleotide bases, sugars or phosphates. The criteria used to define each interaction may be described in detail using `_struct_conn_type.criteria` or a literature reference to the criteria can be given in `_struct_conn_type.reference`.

3.6.7.5.4. *Structural features of monomers*

The data items in these categories are as follows:

(*a*) STRUCT_MON_DETAILS

- `_struct_mon_details.entry_id`
  → `_entry.id`

`_struct_mon_details.prot_cis`
`_struct_mon_details.RSCC`
`_struct_mon_details.RSR`

(*b*) STRUCT_MON_NUCL

- `_struct_mon_nucl.label_alt_id`
  → `_atom_sites_alt.id`
- `_struct_mon_nucl.label_asym_id`
  → `_atom_site.label_asym_id`
- `_struct_mon_nucl.label_comp_id`
  → `_atom_site.label_comp_id`
- `_struct_mon_nucl.label_seq_id`
  → `_atom_site.label_seq_id`

`_struct_mon_nucl.alpha`
`_struct_mon_nucl.auth_asym_id`
  → `_atom_site.auth_asym_id`
`_struct_mon_nucl.auth_comp_id`
  → `_atom_site.auth_comp_id`
`_struct_mon_nucl.auth_seq_id`
  → `_atom_site.auth_seq_id`
`_struct_mon_nucl.beta`
`_struct_mon_nucl.chi1`
`_struct_mon_nucl.chi2`
`_struct_mon_nucl.delta`
`_struct_mon_nucl.details`
`_struct_mon_nucl.epsilon`
`_struct_mon_nucl.gamma`
`_struct_mon_nucl.mean_B_all`
`_struct_mon_nucl.mean_B_base`
`_struct_mon_nucl.mean_B_phos`
`_struct_mon_nucl.mean_B_sugar`
`_struct_mon_nucl.nu0`
`_struct_mon_nucl.nu1`
`_struct_mon_nucl.nu2`
`_struct_mon_nucl.nu3`
`_struct_mon_nucl.nu4`
`_struct_mon_nucl.P`
`_struct_mon_nucl.RSCC_all`
`_struct_mon_nucl.RSCC_base`
`_struct_mon_nucl.RSCC_phos`
`_struct_mon_nucl.RSCC_sugar`
`_struct_mon_nucl.RSR_all`
`_struct_mon_nucl.RSR_base`
`_struct_mon_nucl.RSR_phos`
`_struct_mon_nucl.RSR_sugar`
`_struct_mon_nucl.tau0`
`_struct_mon_nucl.tau1`
`_struct_mon_nucl.tau2`
`_struct_mon_nucl.tau3`
`_struct_mon_nucl.tau4`
`_struct_mon_nucl.taum`
`_struct_mon_nucl.zeta`

(*c*) STRUCT_MON_PROT

- `_struct_mon_prot.label_alt_id`
  → `_atom_sites_alt.id`
- `_struct_mon_prot.label_asym_id`
  → `_atom_site.label_asym_id`
- `_struct_mon_prot.label_comp_id`
  → `_atom_site.label_comp_id`
- `_struct_mon_prot.label_seq_id`
  → `_atom_site.label_seq_id`

`_struct_mon_prot.auth_asym_id`
  → `_atom_site.auth_asym_id`
`_struct_mon_prot.auth_comp_id`
  → `_atom_site.auth_comp_id`
`_struct_mon_prot.auth_seq_id`
  → `_atom_site.auth_seq_id`
`_struct_mon_prot.chi1`
`_struct_mon_prot.chi2`
`_struct_mon_prot.chi3`
`_struct_mon_prot.chi4`
`_struct_mon_prot.chi5`
`_struct_mon_prot.details`
`_struct_mon_prot.RSCC_all`
`_struct_mon_prot.RSCC_main`
`_struct_mon_prot.RSCC_side`
`_struct_mon_prot.RSR_all`
`_struct_mon_prot.RSR_main`
`_struct_mon_prot.RSR_side`
`_struct_mon_prot.mean_B_all`
`_struct_mon_prot.mean_B_main`
`_struct_mon_prot.mean_B_side`
`_struct_mon_prot.omega`
`_struct_mon_prot.phi`
`_struct_mon_prot.psi`

(*d*) STRUCT_MON_PROT_CIS

- `_struct_mon_prot_cis.label_alt_id`
  → `_atom_sites_alt.id`
- `_struct_mon_prot_cis.label_asym_id`
  → `_atom_site.label_asym_id`
- `_struct_mon_prot_cis.label_comp_id`
  → `_atom_site.label_comp_id`
- `_struct_mon_prot_cis.label_seq_id`
  → `_atom_site.label_seq_id`

`_struct_mon_prot_cis.auth_asym_id`
  → `_atom_site.auth_asym_id`
`_struct_mon_prot_cis.auth_comp_id`
  → `_atom_site.auth_comp_id`
`_struct_mon_prot_cis.auth_seq_id`
  → `_atom_site.auth_seq_id`

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

Most macromolecules have complex structures which contain regions of well defined structure and flexible regions that are difficult to model accurately. Overall measures of the quality of a model, such as the standard crystallographic *R* factors, do not represent the local quality of the model. During the development of

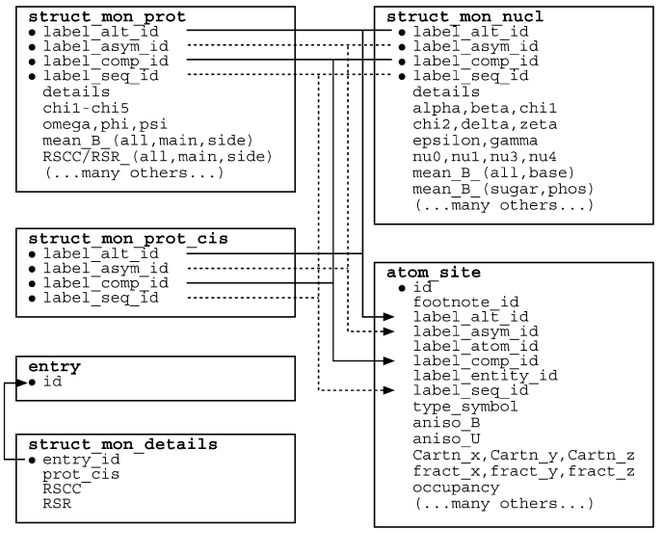


Fig. 3.6.7.11. The family of categories used to describe the structural features of monomers. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

Example 3.6.7.11. *A hypothetical example of the structural features of a single protein residue described with data items in the* STRUCT_MON_PROT *category.*

```
_struct_mon_prot.label_comp_id          ARG
_struct_mon_prot.label_seq_id           35
_struct_mon_prot.label_asym_id          A
_struct_mon_prot.label_alt_id           .
_struct_mon_prot.chi1                 -67.9
_struct_mon_prot.chi2                -174.7
_struct_mon_prot.chi3                 -67.7
_struct_mon_prot.chi4                 -86.3
_struct_mon_prot.chi5                   4.2
_struct_mon_prot.RSCC_all               0.90
_struct_mon_prot.RSR_all                0.18
_struct_mon_prot.mean_B_all            30.0
_struct_mon_prot.mean_B_main           25.0
_struct_mon_prot.mean_B_side           35.1
_struct_mon_prot.omega                180.1
_struct_mon_prot.phi                  -60.3
_struct_mon_prot.psi                  -46.0
```

the mmCIF dictionary, it was found that the biological crystallography community felt that mmCIF should contain data items that allowed the local quality of the model to be recorded: these data items are found in the categories STRUCT_MON_DETAILS, STRUCT_MON_NUCL (for nucleotides), and STRUCT_MON_PROT and STRUCT_MON_PROT_CIS (for proteins). Using these categories, quantities that reflect the local quality of the structure, such as isotropic displacement factors, real-space *R* factors and real-space correlation coefficients, can be given at the monomer and submonomer levels.

In addition, these categories can be used to record the conformation of the structure at the monomer level by listing side-chain torsion angles. These values can be derived from the atom coordinate list, so it would not be common practice to include them in an mmCIF for archiving a structure unless it was to highlight conformations that deviate significantly from expected values (Engh & Huber, 1991). However, there are applications, such as comparative studies across a number of independent determinations of the same structure, where it would be useful to store torsion-angle information without having to recalculate it each time it is needed.

The relationships between the categories used to describe the structural features of monomers are shown in Fig. 3.6.7.11.

Three indicators of the quality of a structure at the local level are included in this version of the dictionary: the mean displacement (*B*) factor, the real-space correlation coefficient (Jones *et al.*, 1991) and the real-space *R* factor (Brändén & Jones, 1990). Other indicators are likely to be added as they become available. In the current version of the dictionary, these metrics can be given at the monomer level, or at the levels of main- and side-chain for proteins, or base, phosphate and sugar for nucleic acids (Altona & Sundaralingam, 1972).

The variables used when calculating real-space correlation coefficients and real-space *R* factors, such as the coefficients used to calculate the map being evaluated or the radii used for including points in a calculation, can be recorded using the data items **_struct_mon_details.RSC** and **_struct_mon_details.RSR**.

These data items are also provided for recording the full conformation of the macromolecule, using a full set of data items for the torsion angles of both proteins and nucleic acids. Although one could use these data items to describe the whole macromolecule, it is more likely that they would be used to highlight regions of the structure that deviate from expected values (Example 3.6.7.11). Deviations from expected values could imply inaccuracies in the model in poorly defined parts of the structure, but in some cases nonstandard torsion angles are found in very well defined regions and are essential to the proper configurations of active sites or ligand binding pockets.

A special case of nonstandard conformation is the occurrence of *cis* peptides in proteins. As the *cis* conformation occurs quite often, the category STRUCT_MON_PROT_CIS is provided so that an explicit list can be made of *cis* peptides. The related data item **_struct_mon_details.prot_cis** allows an author to specify how far a peptide torsion angle can deviate from the expected value of 0.0 and still be considered to be *cis*.

In these categories, properties are listed by residue rather than by individual atom. The only label components needed to identify the residue are ***_alt**, ***_asym**, ***_comp** and ***_seq**. If the author has provided an alternative labelling system, this can also be used. Since the analysis is by individual residue, there is no need to specify symmetry operations that might be needed to move one residue so that it is next to another.

3.6.7.5.5. *Noncrystallographic symmetry*

Data items in these categories are as follows:

(*a*) STRUCT_NCS_ENS

- **_struct_ncs_ens.id**
  _struct_ncs_ens.details
  _struct_ncs_ens.point_group

(*b*) STRUCT_NCS_ENS_GEN

- **_struct_ncs_ens_gen.dom_id_1**
  → **_struct_ncs_dom.id**
- **_struct_ncs_ens_gen.dom_id_2**
  → **_struct_ncs_dom.id**
- **_struct_ncs_ens_gen.ens_id**
  → **_struct_ncs_ens.id**
- **_struct_ncs_ens_gen.oper_id**
  → **_struct_ncs_oper.id**

(*c*) STRUCT_NCS_DOM

- **_struct_ncs_dom.id**
  _struct_ncs_dom.details

(*d*) STRUCT_NCS_DOM_LIM

- **_struct_ncs_dom_lim.beg_label_alt_id**
  → **_atom_sites_alt.id**
- **_struct_ncs_dom_lim.beg_label_asym_id**
  → **_atom_site.label_asym_id**
- **_struct_ncs_dom_lim.beg_label_comp_id**
  → **_atom_site.label_comp_id**

- **_struct_ncs_dom_lim.beg_label_seq_id**
  → **_atom_site.label_seq_id**
- **_struct_ncs_dom_lim.dom_id**
- **_struct_ncs_dom_lim.end_label_alt_id**
  → **_atom_sites_alt.id**
- **_struct_ncs_dom_lim.end_label_asym_id**
  → **_atom_site.label_asym_id**
- **_struct_ncs_dom_lim.end_label_comp_id**
  → **_atom_site.label_comp_id**
- **_struct_ncs_dom_lim.end_label_seq_id**
  → **_atom_site.label_seq_id**

_struct_ncs_dom_lim.beg_auth_asym_id
→ **_atom_site.auth_asym_id**
_struct_ncs_dom_lim.beg_auth_comp_id
→ **_atom_site.auth_comp_id**
_struct_ncs_dom_lim.beg_auth_seq_id
→ **_atom_site.auth_seq_id**
_struct_ncs_dom_lim.end_auth_asym_id
→ **_atom_site.auth_asym_id**
_struct_ncs_dom_lim.end_auth_comp_id
→ **_atom_site.auth_comp_id**
_struct_ncs_dom_lim.end_auth_seq_id
→ **_atom_site.auth_seq_id**

(*e*) STRUCT_NCS_OPER

- **_struct_ncs_oper.id**

_struct_ncs_oper.code
_struct_ncs_oper.details
_struct_ncs_oper.matrix[1][1]
_struct_ncs_oper.matrix[1][2]
_struct_ncs_oper.matrix[1][3]
_struct_ncs_oper.matrix[2][1]
_struct_ncs_oper.matrix[2][2]
_struct_ncs_oper.matrix[2][3]
_struct_ncs_oper.matrix[3][1]
_struct_ncs_oper.matrix[3][2]
_struct_ncs_oper.matrix[3][3]
_struct_ncs_oper.vector[1]
_struct_ncs_oper.vector[2]
_struct_ncs_oper.vector[3]

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

Biological macromolecular complexes may be built from domains related by symmetry transformations other than those arising from the crystal lattice symmetry. These domains are not necessarily discrete molecular entities: they may be composed of one or more segments of a single polypeptide or nucleic acid chain, of segments from more than one chain, or of small-molecule components of the structure. The categories above allow the distinct domains that participate in ensembles of structural elements related by noncrystallographic symmetry to be listed and described in detail. The relationships between categories used to describe noncrystallographic symmetry are shown in Fig. 3.6.7.12.

In the mmCIF model of noncrystallographic symmetry, the highest level of organization is the ensemble, which corresponds to the complete symmetry-related aggregate (*e.g.* tetramer, icosahedron). An identifier is given to the ensemble using the data item **_struct_ncs_ens.id**.

The symmetry-related elements within the ensemble are referred to as domains. The elements of structure that are to be considered part of the domain are specified using the data items in the STRUCT_NCS_DOM and STRUCT_NCS_DOM_LIM categories. By using the STRUCT_NCS_DOM_LIM data items appropriately, domains can be defined to include ranges of polypeptide chain or nucleic acid strand, bound ligands or cofactors, or even bound solvent molecules. Note that the category keys for STRUCT_NCS_DOM_LIM include the domain ID and the range specifiers. Thus a single domain may be composed of any number of ranges of elements.

Finally, the ensemble is generated from the domains using the rotation matrix and translation vector specified by data items in

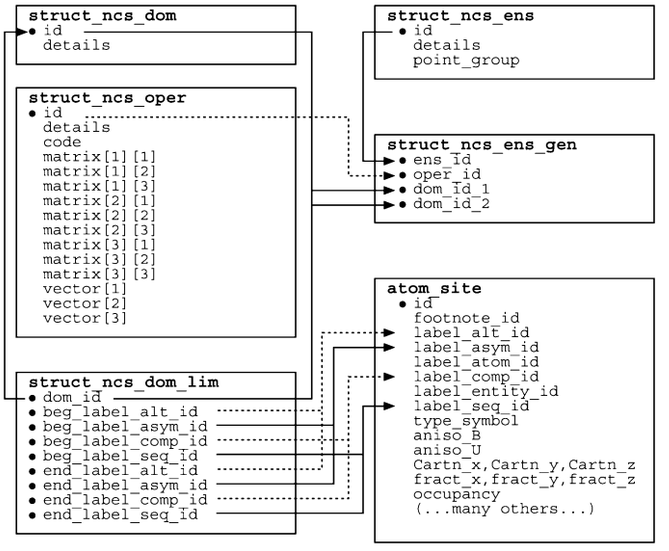


Fig. 3.6.7.12. The family of categories used to describe noncrystallographic symmetry. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

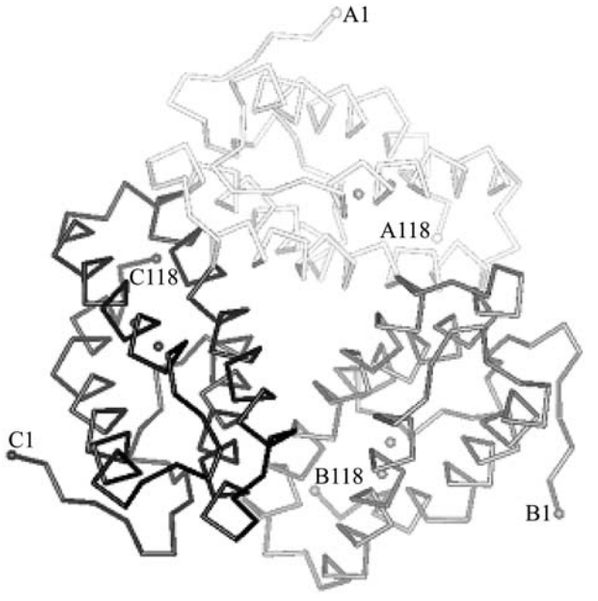


Fig. 3.6.7.13. Noncrystallographic symmetry in the structure of trimeric haemerythrin (PDB 1HR3) to be described with data items in the STRUCT_NCS_ENS, STRUCT_NCS_ENS_GEN, STRUCT_NCS_DOM and STRUCT_NCS_DOM_LIM categories.

the STRUCT_NCS_OPER category, which are referenced by the data items in the STRUCT_NCS_ENS_GEN category. There are data items appropriate for two common methods of describing noncrystallographic symmetry:

(1) In the first method, the coordinate list includes all copies of domains related by noncrystallographic symmetry and the aim is to describe the relationships between domains in the ensemble; in this case the data items in STRUCT_NCS_ENS_GEN specify a pair of domains and reference the appropriate operator in STRUCT_NCS_OPER. This method is indicated by giving the data item **_struct_ncs_oper.code** the value `given`.

(2) In the second method, the coordinate list contains only one copy of the domain and the aim is to generate the entire ensemble; in this case the data items in STRUCT_NCS_ENS_GEN

Example 3.6.7.12. *Noncrystallographic symmetry in the structure of trimeric haemerythrin (PDB 1HR3) described with data items in the* STRUCT_NCS_ENS, STRUCT_NCS_ENS_GEN, STRUCT_NCS_DOM *and* STRUCT_NCS_DOM_LIM *categories. For brevity, the data items in the* STRUCT_NCS_OPER *category are not shown.*

```
_struct_ncs_ens.id                    trimer
_struct_ncs_ens.point_group           3

loop_
_struct_ncs_ens_gen.ens_id
_struct_ncs_ens_gen.dom_id_1
_struct_ncs_ens_gen.dom_id_2
_struct_ncs_ens_gen.oper_id
 trimer  chain_A  chain_B  1
 trimer  chain_A  chain_C  2

loop_
_struct_ncs_dom.id
 chain_A    chain_B    chain_C

loop_
_struct_ncs_dom_lim.dom_id
_struct_ncs_dom_lim.beg_label_asym_id
_struct_ncs_dom_lim.beg_label_comp_id
_struct_ncs_dom_lim.beg_label_seq_id
_struct_ncs_dom_lim.beg_label_alt_id
_struct_ncs_dom_lim.end_label_asym_id
_struct_ncs_dom_lim.end_label_comp_id
_struct_ncs_dom_lim.end_label_seq_id
_struct_ncs_dom_lim.end_label_alt_id
 chain_A  A  ala  1  .  A  ala  118  .
 chain_B  B  ala  1  .  B  ala  118  .
 chain_C  C  ala  1  .  C  ala  118  .
```

specify a pair of domains and reference the appropriate operator in STRUCT_NCS_OPER, but now the data item **_struct_ncs_oper.code** is given the value `generate`.

Noncrystallographic symmetry in a trimeric molecule is shown in Fig. 3.6.7.13 and described in Example 3.6.7.12.

3.6.7.5.6. *External databases*

The data items in these categories are as follows:

(*a*) STRUCT_REF

```
• _struct_ref.id
  _struct_ref.biol_id
        → _struct_biol.id
  _struct_ref.db_code
  _struct_ref.db_name
  _struct_ref.details
  _struct_ref.entity_id
        → _entity.id
  _struct_ref.seq_align
  _struct_ref.seq_dif
```

(*b*) STRUCT_REF_SEQ

```
• _struct_ref_seq.align_id
  _struct_ref_seq.db_align_beg
  _struct_ref_seq.db_align_end
  _struct_ref_seq.details
  _struct_ref_seq.ref_id
        → _struct_ref.id
  _struct_ref_seq.seq_align_beg
        → _entity_poly_seq.num
  _struct_ref_seq.seq_align_end
        → _entity_poly_seq.num
```

(*c*) STRUCT_REF_SEQ_DIF

```
• _struct_ref_seq_dif.align_id
        → _struct_ref_seq.align_id
• _struct_ref_seq_dif.seq_num
        → _entity_poly_seq.num
  _struct_ref_seq_dif.db_mon_id
        → _chem_comp.id
  _struct_ref_seq_dif.details
  _struct_ref_seq_dif.mon_id
        → _chem_comp.id
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

Data items in the STRUCT_REF category allow the author of an mmCIF to provide references to information in external databases that is relevant to the entities or biological units described in the mmCIF. For example, the database entry for a protein or nucleic acid sequence could be referenced and any differences between the sequence of the macromolecule whose structure is reported in the mmCIF and the sequence of the related entry in the external database can be recorded. Alternatively, references to external database entries can be used to record the relationship of the structure reported in the mmCIF to structures already reported in the literature, for example by referring to previously determined structures of the same or a similar protein, or to a small-molecule structure determination of a bound inhibitor or cofactor. STRUCT_REF data items are not intended to be used to reference a database entry for the structure in the mmCIF itself (this would be the role of data items in the DATABASE_2 category), but it would not be formally incorrect to do so.

When the data items in these categories are used to provide references to external database entries describing the sequence of a polymer, data items from all three categories could be used. The value of the data item **_struct_ref.seq_align** is used to indicate whether the correspondence between the sequence of the entity or biological unit in the mmCIF and the sequence in the related external database entry is `complete` or `partial`. If the value is `partial`, the region (or regions) of the alignment may be identified using data items in the STRUCT_REF_SEQ category. Comments on the alignment may be given in **_struct_ref_seq.details** (Example 3.6.7.13).

The value of the data item **_struct_ref.seq_dif** is used to indicate whether the two sequences contain point differences. If the value is `yes`, the differences may be identified and annotated using data items in the STRUCT_REF_SEQ_DIF category. Comments on specific point differences may be recorded in **_struct_ref_seq_dif.details**.

Example 3.6.7.13. *The relationship of the sequence of the protein PDB 5HVP to a sequence in an external database described with data items in the* STRUCT_REF *and* STRUCT_REF_SEQ *categories.*

```
loop_
  _struct_ref.id
  _struct_ref.biol_id
  _struct_ref.entity_id
  _struct_ref.db_name
  _struct_ref.db_code
  _struct_ref.seq_align
  _struct_ref.seq_dif
 seq_pdb     1  .  PDB       5HVP       .          .
 seq_genbank .  1  GenBank   AAG30358   complete   yes

loop_
  _struct_ref_seq.align_id
  _struct_ref_seq.ref_id
  _struct_ref_seq.seq_align_beg
  _struct_ref_seq.seq_align_end
  _struct_ref_seq.db_align_beg
  _struct_ref_seq.db_align_end
  _struct_ref_seq.details
    align_seq_pdb_genbank seq_genbank 1 99 24 122
; The genbank reference is to the sequence of
  residues 1-376 of the viral pol 1 polypeptide;
  the protease is proteolytically released from
  this precursor during viral maturation.
;
```

References do not have to be to entries in databases of sequences: any external database can be referenced. For other kinds of databases, only the data items in the STRUCT_REF category would usually be used. The element of the structure that is referenced could be either an entity or a biological unit, that is, either a building block of the structure or a structurally meaningful assembly of those building blocks. Since the identification of the part of the structure being linked to an entry in an external database can be made using either `_struct_ref.biol_id` or `_struct_ref.entity_id`, and since any part of the structure could be linked to any number of entries in external databases, the data item `_struct_ref.id` was introduced as the category key.

3.6.7.5.7. *β-sheets*

Data items in these categories are as follows:

(*a*) STRUCT_SHEET

```
• _struct_sheet.id
  _struct_sheet.details
  _struct_sheet.number_strands
  _struct_sheet.type
```

(*b*) STRUCT_SHEET_TOPOLOGY

```
• _struct_sheet_topology.range_id_1
        → _struct_sheet_range.id
• _struct_sheet_topology.range_id_2
        → _struct_sheet_range.id
• _struct_sheet_topology.sheet_id
        → _struct_sheet.id
  _struct_sheet_topology.offset
  _struct_sheet_topology.sense
```

(*c*) STRUCT_SHEET_RANGE

```
• _struct_sheet_range.id
• _struct_sheet_range.sheet_id
        → _struct_sheet.id
  _struct_sheet_range.beg_label_asym_id
        → _struct_asym.id
  _struct_sheet_range.beg_label_comp_id
        → _chem_comp.id
  _struct_sheet_range.beg_label_seq_id
        → _atom_site.label_seq_id
  _struct_sheet_range.end_label_asym_id
        → _struct_asym.id
  _struct_sheet_range.end_label_comp_id
        → _chem_comp.id
  _struct_sheet_range.end_label_seq_id
        → _atom_site.label_seq_id
  _struct_sheet_range.beg_auth_asym_id
        → _atom_site.auth_atom_id
  _struct_sheet_range.beg_auth_comp_id
        → _atom_site.auth_comp_id
  _struct_sheet_range.beg_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_sheet_range.end_auth_asym_id
        → _atom_site.auth_atom_id
  _struct_sheet_range.end_auth_comp_id
        → _atom_site.auth_comp_id
  _struct_sheet_range.end_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_sheet_range.symmetry
```

(*d*) STRUCT_SHEET_ORDER

```
• _struct_sheet_order.range_id_1
        → _struct_sheet_range.id
• _struct_sheet_order.range_id_2
        → _struct_sheet_range.id
• _struct_sheet_order.sheet_id
        → _struct_sheet.id
  _struct_sheet_order.offset
  _struct_sheet_order.sense
```

(*e*) STRUCT_SHEET_HBOND

```
• _struct_sheet_hbond.range_id_1
        → _struct_sheet_range.id
• _struct_sheet_hbond.range_id_2
        → _struct_sheet_range.id
• _struct_sheet_hbond.sheet_id
        → _struct_sheet.id
  _struct_sheet_hbond.range_1_beg_label_atom_id
        → _atom_site.label_atom_id
  _struct_sheet_hbond.range_1_beg_label_seq_id
        → _atom_site.label_seq_id
  _struct_sheet_hbond.range_1_end_label_atom_id
        → _atom_site.label_atom_id
  _struct_sheet_hbond.range_1_end_label_seq_id
        → _atom_site.label_seq_id
  _struct_sheet_hbond.range_2_beg_label_atom_id
        → _atom_site.label_atom_id
  _struct_sheet_hbond.range_2_beg_label_seq_id
        → _atom_site.label_seq_id
  _struct_sheet_hbond.range_2_end_label_atom_id
        → _atom_site.label_atom_id
  _struct_sheet_hbond.range_2_end_label_seq_id
        → _atom_site.label_seq_id
  _struct_sheet_hbond.range_1_beg_auth_atom_id
        → _atom_site.auth_atom_id
  _struct_sheet_hbond.range_1_beg_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_sheet_hbond.range_1_end_auth_atom_id
        → _atom_site.auth_atom_id
  _struct_sheet_hbond.range_1_end_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_sheet_hbond.range_2_beg_auth_atom_id
        → _atom_site.auth_atom_id
  _struct_sheet_hbond.range_2_beg_auth_seq_id
        → _atom_site.auth_seq_id
  _struct_sheet_hbond.range_2_end_auth_atom_id
        → _atom_site.auth_atom_id
  _struct_sheet_hbond.range_2_end_auth_seq_id
        → _atom_site.auth_seq_id
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

Different methods of describing β-sheets are in widespread use. The mmCIF dictionary provides data items for two methods and it is anticipated that future versions of the dictionary could cover others. The model used in the STRUCT_SHEET_TOPOLOGY category is the simpler of the two. It is a convenient shorthand for describing the topology, but it does not provide details about strand registration and it is not suitable for describing sheets that contain strands from more than one polypeptide. A more general model is provided by the linked data items in the STRUCT_SHEET_RANGE, STRUCT_SHEET_ORDER and STRUCT_SHEET_HBOND categories. For both methods of representing β-sheets, data items in the parent category STRUCT_SHEET can be used to provide an identifier for each sheet, a free-text description of its type, the number of participating strands and a free-text description of any peculiar aspects of the sheet. The relationships between categories used to describe β-sheets are shown in Fig. 3.6.7.14.

In the description of β-sheet topology based on the STRUCT_SHEET_TOPOLOGY category, the strand that occurs first in the polypeptide chain is numbered 1. Subsequent strands are described by their position in the sheet relative to the previous strand (+1, −3 *etc.*) and by their orientation relative to the previous strand (parallel or antiparallel).

While writing this chapter, a few errors in the mmCIF dictionary were discovered. The use of `_struct_sheet_topology.range_id_1` and `*_2` as pointers to the residues participating in β-sheets is one; the correct data items should be `_struct_sheet_topology.comp_id_1` and `*_2`, and these data items should be pointers to `_atom_site.label_comp_id`. This error will be corrected in future versions of the dictionary. As the data model encoded in the current version of the dictionary is incorrect, no example of its use is given.

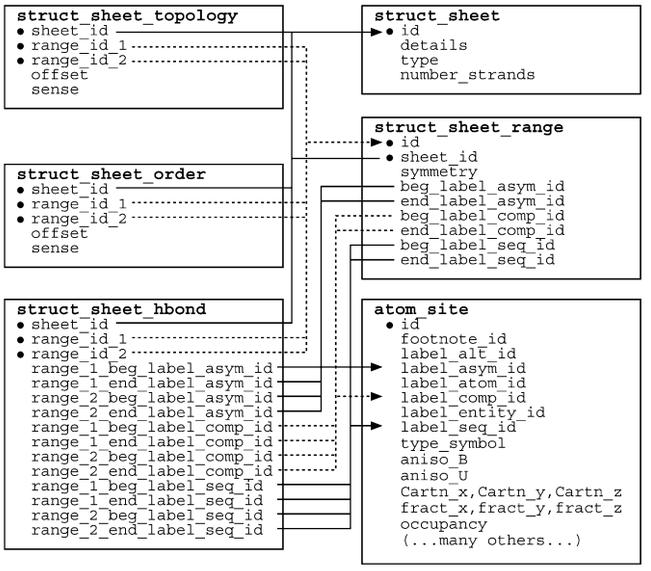


Fig. 3.6.7.14. The family of categories used to describe β-sheets. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

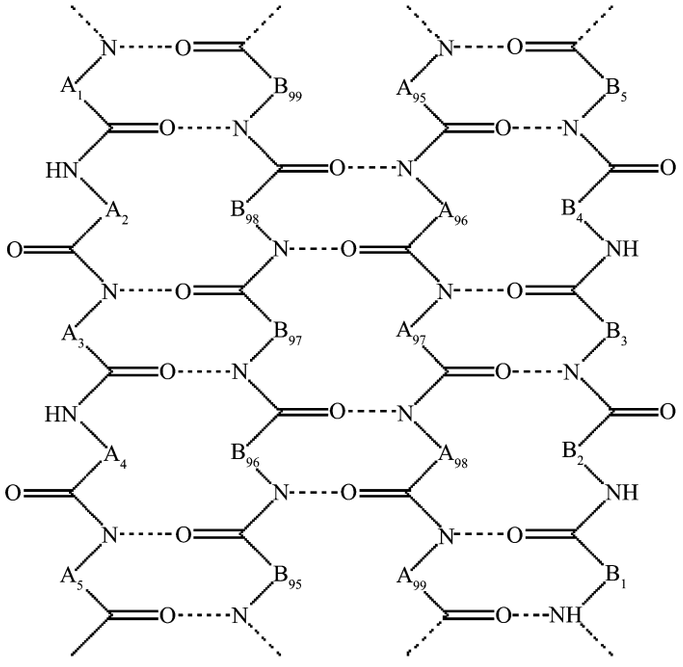


Fig. 3.6.7.15. A hypothetical β-sheet to be described with data items in the STRUCT_SHEET, STRUCT_SHEET_ORDER, STRUCT_SHEET_RANGE and STRUCT_SHEET_HBOND categories. Note that the strands come from two different polypeptides, labelled A and B.

In the more detailed and more general method for describing β-sheets, data items in the STRUCT_SHEET_RANGE category specify the range of residues that form strands in the sheet, data items in the STRUCT_SHEET_ORDER category specify the relative pairwise orientation of strands and data items in the STRUCT_SHEET_HBOND category provide details of specific hydrogen-bonding interactions between strands (see Fig. 3.6.7.15 and Example 3.6.7.14). Note that the specifiers for the strand ranges include the amino acid (**`*_comp_id`** and **`*_seq_id`**), the chain (**`*_asym_id`**) and a symmetry code (**`_struct_sheet_range.symmetry`**). Thus sheets that are composed of strands from more than one polypeptide chain

Example 3.6.7.14. *A hypothetical β-sheet described with data items in the STRUCT_SHEET, STRUCT_SHEET_ORDER, STRUCT_SHEET_RANGE and STRUCT_SHEET_HBOND categories.*

```
loop_
  _struct_sheet.id
  _struct_sheet.number_strands
    S1   4

loop_
  _struct_sheet_order.sheet_id
  _struct_sheet_order.range_id_1
  _struct_sheet_order.range_id_2
  _struct_sheet_order.sense
    S1 1 2 anti-parallel
    S1 2 3 anti-parallel
    S1 3 4 anti-parallel
    S2 1 2 anti-parallel

loop_
  _struct_sheet_range.sheet_id
  _struct_sheet_range.id
  _struct_sheet_range.beg_label_comp_id
  _struct_sheet_range.beg_label_asym_id
  _struct_sheet_range.beg_label_seq_id
  _struct_sheet_range.end_label_comp_id
  _struct_sheet_range.end_label_asym_id
  _struct_sheet_range.end_label_seq_id
    S1 1 PRO A 1  LEU A 5
    S1 2 CYS B 95 PHE B 99
    S1 3 CYS A 95 PHE A 99
    S1 4 PRO B 1  LEU B 5

loop_
  _struct_sheet_hbond.sheet_id
  _struct_sheet_hbond.range_id_1
  _struct_sheet_hbond.range_id_2
  _struct_sheet_hbond.range_1_beg_label_atom_id
  _struct_sheet_hbond.range_1_beg_label_seq_id
  _struct_sheet_hbond.range_2_beg_label_atom_id
  _struct_sheet_hbond.range_2_beg_label_seq_id
    S1 1 2 A 3  O 97
    S1 2 3 B 98 O 96
    S1 3 4 A 97 O 3
```

or from polypeptides in more than one asymmetric unit can be described.

It is conventional to assign the number 1 to an outermost strand. The choice of which outermost strand to number as 1 is arbitrary, but would usually be the strand encountered first in the amino-acid sequence. The remaining strands are then numbered sequentially across the sheet.

In some simple cases, the complete hydrogen bonding of the sheet could be inferred from the strand-range pairings and the relationship between the strands (parallel or antiparallel). However, in most cases it is necessary to specify at least one hydrogen bond between adjacent strands in order to establish the registration. The data items in the STRUCT_SHEET_HBOND category can be used to do this. Hydrogen bonds also need to be specified precisely when a sheet contains a nonstandard feature such as a β-bulge. This is a case where it is sufficient to specify a single hydrogen-bonding interaction to establish the registration; here only the **`*_beg_*`** or **`*_end_*`** data items need to be used to reference the atom-label components. However, it is preferable, wherever possible, to specify the initial and final atoms of the two ranges participating in the hydrogen bonding.

3.6.7.5.8. *Molecular sites*

The data items in these categories are as follows:

(*a*) STRUCT_SITE

- **`_struct_site.id`**
  `_struct_site.details`

(*b*) STRUCT_SITE_KEYWORDS

```
• _struct_site_keywords.site_id
        → _struct_site.id
• _struct_site_keywords.text
```

(*c*) STRUCT_SITE_GEN

```
• _struct_site_gen.id
• _struct_site_gen.site_id
        → _struct_site.id
  _struct_site_gen.details
  _struct_site_gen.label_alt_id
        → _atom_sites_alt.id
  _struct_site_gen.label_asym_id
        → _atom_site.label_asym_id
  _struct_site_gen.label_atom_id
        → _chem_comp_atom.atom_id
  _struct_site_gen.label_comp_id
        → _atom_site.label_atom_id
  _struct_site_gen.label_seq_id
        → _atom_site.label_seq_id
  _struct_site_gen.auth_asym_id
        → _atom_site.auth_asym_id
  _struct_site_gen.auth_atom_id
        → _atom_site.auth_atom_id
  _struct_site_gen.auth_comp_id
        → _atom_site.auth_comp_id
  _struct_site_gen.auth_seq_id
        → _atom_site.auth_seq_id
  _struct_site_gen.symmetry
```

(*d*) STRUCT_SITE_VIEW

```
• _struct_site_view.id
  _struct_site_view.details
  _struct_site_view.rot_matrix[1][1]
  _struct_site_view.rot_matrix[1][2]
  _struct_site_view.rot_matrix[1][3]
  _struct_site_view.rot_matrix[2][1]
  _struct_site_view.rot_matrix[2][2]
  _struct_site_view.rot_matrix[2][3]
  _struct_site_view.rot_matrix[3][1]
  _struct_site_view.rot_matrix[3][2]
  _struct_site_view.rot_matrix[3][3]
  _struct_site_view.site_id
        → _struct_site.id
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

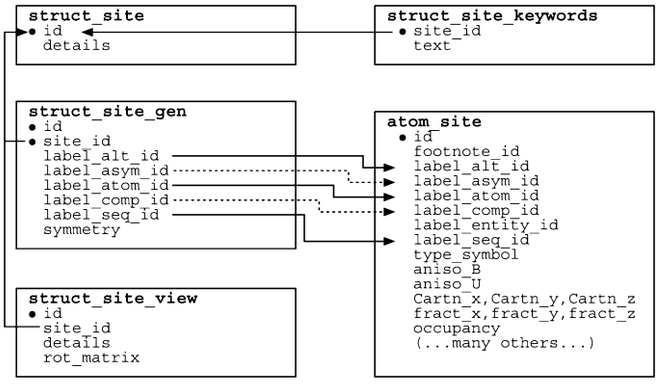


Fig. 3.6.7.16. The family of categories used to describe molecular sites. Boxes surround categories of related data items. Data items that serve as category keys are preceded by a bullet (•). Lines show relationships between linked data items in different categories with arrows pointing at the parent data items.

Substrate-binding sites, active sites, metal coordination sites and any other sites of interest may be described using data items in a collection of categories descending from STRUCT_SITE. These categories are intended to enable the author to generate views of molecular sites that could be used as figures in a report describing the structure or to enable a database to store standard views of common molecular sites (*e.g.* ATP-binding sites or the coordination of a calcium atom). The relationships between categories used to describe structural sites are shown in Fig. 3.6.7.16.

An identifier for each site that an author wishes to describe is given using `_struct_site.id` and the site can be described using `_struct_site.details`.

Keywords can be given for each site using data items in the STRUCT_SITE_KEYWORD category. Because keywords can be given at many levels of the mmCIF description of a structure, it may be worth duplicating the most significant higher-level keywords at this level to ensure that the site is detected in all search strategies.

The structural elements that generate each molecular site can be specified using data items in the STRUCT_SITE_GEN category. 'Structural elements' in this sense may be at any level of detail in the structure: single atoms, complete amino acids or nucleotides, or elements of secondary, tertiary or quaternary structure. Therefore the labels for each element may include, as required, the relevant `*_alt`, `*_asym`, `*_atom`, `*_comp` or `*_seq` parts of atom or residue identifiers. If the author has used an alternative labelling scheme, this can also be used. Noteworthy features of a structural element that forms part of the site can be described using the data item `_struct_site_gen.details`. Any crystallographic symmetry operations that are needed to form the site can be given using `_struct_site_gen.symmetry`.

Data items in the STRUCT_SITE_VIEW category allow the author to specify an orientation of the molecular site that gives a useful view of the components. The comments given in `_struct_site_view.details` could be used as a figure caption if the view is intended for use as a figure in a report.

Example 3.6.7.15 illustrates the use of these categories for describing a DNA binding site.

Example 3.6.7.15. *A DNA binding site with an intercalated drug (NDB DDF040) described with data items in the STRUCT_SITE, STRUCT_SITE_KEYWORDS, STRUCT_SITE_GEN and STRUCT_SITE_VIEW categories.*

```
loop_
_struct_site.id
_struct_site.details
   B1  'Binding at TG/AC Step 1'

loop_
_struct_site_keywords.site_id
_struct_site_keywords.text
   B1  'Intercalation complex'

loop_
_struct_site_gen.id
_struct_site_gen.site_id
_struct_site_gen.label_asym_id
_struct_site_gen.label_comp_id
_struct_site_gen.label_seq_id
_struct_site_gen.symmetry
   1  B1  A  T    1  1_555
   2  B1  A  G    2  1_555
   3  B1  A  C    5  8_555
   4  B1  A  A    6  8_555
   5  B1  D  DM2  .  8_555

loop_
_struct_site_view.id
_struct_site_view.site_id
_struct_site_view.details
_struct_site_view.rot_matrix[1][1]
_struct_site_view.rot_matrix[1][2]
# - - - abbreviated - - -
_struct_site_view.rot_matrix[3][3]
   View1  B1
  'View along the base-pair plane'
   0.133  0.922  . . . . . -0.172
```

3.6.7.6. Crystal symmetry

The categories describing symmetry are as follows:

SYMMETRY group
- SYMMETRY
- SYMMETRY_EQUIV
- SPACE_GROUP
- SPACE_GROUP_SYMOP

Data items in the SYMMETRY category are used to give details about the crystallographic symmetry. The equivalent positions for the space group are listed using data items in the SYMMETRY_EQUIV category. These categories are used in the same way in the core CIF and mmCIF dictionaries, and Section 3.2.4.4 can be consulted for details.

The current version of the mmCIF dictionary includes the SPACE_GROUP categories that were derived from the symmetry CIF dictionary (Chapter 3.8) and included in version 2.3 of the core CIF dictionary. At the time of writing, macromolecular applications have not yet begun to make use of these new categories.

Data items in these categories are as follows:

(*a*) SYMMETRY
- **_symmetry.entry_id**
  → **_entry.id**
- **_symmetry.cell_setting**
- **_symmetry.Int_Tables_number**
- **_symmetry.space_group_name_Hall**
- **_symmetry.space_group_name_H-M**

(*b*) SYMMETRY_EQUIV
- **_symmetry_equiv.id** (∼ **_symmetry_equiv_pos_site_id**)
- **_symmetry_equiv.pos_as_xyz**

(*c*) SPACE_GROUP
- **_space_group.id**
- **_space_group.crystal_system**
- **_space_group.IT_number**
- **_space_group.name_H-M_alt**
- **_space_group.name_Hall**

(*d*) SPACE_GROUP_SYMOP
- **_space_group_symop.id**
- **_space_group_symop.operation_xyz**
- **_space_group_symop.sg_id**

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_) except where indicated by the ∼ symbol.

The data item **_symmetry.entry_id** has been added to the SYMMETRY category to provide the formal category key required by the DDL2 data model.

3.6.7.7. Bond-valence information

The categories describing bond valences are as follows:

VALENCE group
- VALENCE_PARAM
- VALENCE_REF

These categories were introduced into version 2.2 of the core CIF dictionary to provide the information about bond valences required in inorganic crystallography. They appear in the mmCIF dictionary only for full compatibility with the core dictionary.

Data items in these categories are as follows:

(*a*) VALENCE_PARAM
- **_valence_param.atom_1**
- **_valence_param.atom_1_valence**
- **_valence_param.atom_2**
- **_valence_param.atom_2_valence**
- **_valence_param.B**
- **_valence_param.details**
- **_valence_param.id**
- **_valence_param.ref_id**
  → **_valence_ref.id**
- **_valence_param.Ro**

(*b*) VALENCE_REF
- **_valence_ref.id**
- **_valence_ref.reference**

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_).

Information about the use of these data items in the core CIF dictionary is given in Section 3.2.4.5.

3.6.8. Publication

The results of the determination of the crystal structure of a biological macromolecule might be published in an academic journal and/or deposited in a structural database. The data items in the core CIF dictionary cover most of the requirements for constructing an article for publication from an mmCIF and the many well defined data fields in mmCIF allow an extensively annotated record of the structure to be deposited in a database. However, the formalism of two of the core CIF categories for publication did not fit the relational database model of mmCIF, so new categories were required. The core CIF category COMPUTING, which is used to list the programs used to determine the structure, is replaced by the mmCIF category SOFTWARE, and the core CIF category DATABASE, which is used to identify the records associated with the structure in various databases, is replaced by the mmCIF category DATABASE_2.

The category groups discussed here are: the CITATION group, which is used to give citations to the literature (Section 3.6.8.1); the COMPUTING group, which is used to cite software (Section 3.6.8.2); the DATABASE group for citing related database entries (Section 3.6.8.3), which includes a group of categories used to ensure compatibility with specific database records in the Protein Data Bank (Section 3.6.8.3.2); journal administration categories that might be used by a publisher (Section 3.6.8.4.1); and the PUBL family of categories used to store the text of an article for publication (Section 3.6.8.4.2).

3.6.8.1. Literature citations

The categories describing literature citations are as follows:

CITATION group
- CITATION
- CITATION_AUTHOR
- CITATION_EDITOR

Data items in these categories are as follows:

(*a*) CITATION
- **_citation.id**
- **_citation.abstract**
- **_citation.abstract_id_CAS**
- **_citation.book_id_ISBN**
- **_citation.book_publisher**
- **_citation.book_publisher_city**
- **_citation.book_title**
- **_citation.coordinate_linkage**
- **_citation.country**
- **_citation.database_id_CSD**
- **_citation.database_id_Medline**
- **_citation.journal_abbrev**
- **_citation.journal_full**
- **_citation.journal_id_ASTM**
- **_citation.journal_id_CSD**
- **_citation.journal_id_ISSN**
- **_citation.journal_issue**
- **_citation.journal_volume**
- **_citation.language**
- **_citation.page_first**

```
  _citation.page_last
  _citation.details (~ _citation_special_details)
  _citation.title
  _citation.year
```

(*b*) CITATION_AUTHOR

```
• _citation_author.citation_id
        → _citation.id
  _citation_author.name
  _citation_author.ordinal
```

(*c*) CITATION_EDITOR

```
• _citation_editor.citation_id
        → _citation.id
  _citation_editor.name
  _citation_editor.ordinal
```

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (.) *to an underscore* (_).

The original core CIF dictionary contained the data item `_publ_section_references` for citations of journal articles, book chapters and monographs. The authors of the mmCIF dictionary felt that a more detailed and structured approach to literature citations was required. This is provided by the mmCIF categories CITATION, CITATION_AUTHOR and CITATION_EDITOR. These categories were subsequently included in the core CIF dictionary and are used in the same way in both dictionaries. Section 3.2.5.1 may be consulted for details. Although `_publ.section_references` remains a valid mmCIF data item, it is expected that the CITATION, CITATION_AUTHOR and CITATION_EDITOR categories will be used for literature citations in mmCIFs.

3.6.8.2. Citation of software packages

The categories describing software citations are as follows:

COMPUTING group
  COMPUTING
  SOFTWARE

It is expected that citations of software packages in an mmCIF will be made using data items in the SOFTWARE category. However, in some cases, a particular publisher or database may require that this information is given using data items in the COMPUTING category instead (see Section 3.2.5.2 for details).

Data items in these categories are as follows:

(*a*) COMPUTING

```
• _computing.entry_id
        → _entry.id
  _computing.cell_refinement
  _computing.data_collection
  _computing.data_reduction
  _computing.molecular_graphics
  _computing.publication_material
  _computing.structure_refinement
  _computing.structure_solution
```

(*b*) SOFTWARE

```
• _software.name
• _software.version
  _software.citation_id
        → _citation.id
  _software.classification
  _software.compiler_name
  _software.compiler_version
  _software.contact_author
  _software.contact_author_email
  _software.date
  _software.dependencies
  _software.description
  _software.hardware
  _software.language
  _software.location
  _software.mods
  _software.os
  _software.os_version
  _software.type
```

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (.) *to an underscore* (_).

The data item `_computing.entry_id` has been added to the COMPUTING category to provide the formal category key required by the DDL2 data model.

The data items in the SOFTWARE category are used to cite the software packages used in the structure analysis. The software can be described in great detail if necessary. However, for most applications a small subset of these data items, for example just `_software.name` and `_software.version`, could be used (see Example 3.6.8.1).

Most data items in the SOFTWARE category are self-explanatory, but a few require further comment. The data item `_software.citation_id` provides a way to link the details of a program to the citation of an article in the literature that describes the program; this data item must match a value of `_citation.id` in the CITATION category. The name and e-mail address of the author of the software can also be given using `_software.contact_author` and `_software.contact_author_email`, respectively. (This may be the original author or someone who subsequently modifies or maintains the software; these data items would generally refer to the person most closely associated with the maintenance of the code at the time it was used.) The release date of the software may be recorded in `_software.date`. As far as possible, the date should be that of the version recorded in `_software.version`. The data item `_software.location` may be used to supply a URL from which the software may be downloaded or where it is described in detail.

3.6.8.3. Citation of related database entries

Categories describing related database entries are as follows:

DATABASE group
Related database entries (§3.6.8.3.1)
  DATABASE
  DATABASE_2

Example 3.6.8.1. *The refinement program Prolsq described with data items in the SOFTWARE category.*

```
_software.name
_software.version
_software.date
_software.type
_software.contact_author
_software.contact_author_email
_software.location
_software.classification
_software.citation_id
_software.language
_software.compiler_name
_software.compiler_version
_software.hardware
_software.os
_software.os_version
_software.dependencies
_software.mods
_software.description
   Prolsq    unknown  .    program
    'Wayne A. Hendrickson'  ?
    'ftp://rosebud.sdsc.edu/pub/sdsc/xtal/CCP4/ccp4/'
    refinement ref5  Fortran
  'Convex Fortran' v8.0 'Convex C220' ConvexOS  v10.1
    'Requires that Protin be run first'    optimized
    'restrained least-squares refinement'
```

Compatibility with PDB format files (§3.6.8.3.2)

DATABASE_PDB_CAVEAT
DATABASE_PDB_MATRIX
DATABASE_PDB_REMARK
DATABASE_PDB_REV
DATABASE_PDB_REV_RECORD
DATABASE_PDB_TVECT

The purpose of entries in the DATABASE category group is to provide pointers that link the mmCIF to all database entries that result from the deposition of the file. For mmCIF, the relevant category is DATABASE_2, which replaces the DATABASE category of the core dictionary.

Note the distinction between the database pointers provided here and those in the STRUCT_REF family of categories. The latter are intended to provide links to external database entries for any aspect of any subset of the structure that the author may wish to record, including previous determinations of the same structure, other structures containing the same ligand or references to the sequence(s) of the macromolecule(s) in sequence databases. In contrast, the links provided in DATABASE_2 refer to the entire contents of the mmCIF and are designed to cover situations in which the entire file is deposited in more than one database (for example, in the PDB and in a database for protein kinases).

3.6.8.3.1. *Related database entries*

Data items in these categories are as follows:

(*a*) DATABASE

- **_database.entry_id**
  → **_entry.id**
  _database.code_CAS
  _database.code_CSD
  _database.code_ICSD
  _database.code_MDF
  _database.code_NBS
  _database.code_PDB
  _database.code_PDF
  _database.code_depnum_ccdc_archive
  _database.code_depnum_ccdc_fiz
  _database.code_depnum_ccdc_journal
  _database.CSD_history
  _database.journal_ASTM
  _database.journal_CSD

(*b*) DATABASE_2

- **_database_2.database_id**
- **_database_2.database_code**

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop* (**.**) *to an underscore* (_).

The DATABASE category is retained in the mmCIF dictionary, but only for consistency with the core dictionary.

The role of the data items in the DATABASE_2 category is to store identifiers assigned by one or more databases to the structure described in the mmCIF. In the data model used in the core CIF dictionary, each database has an individual data item. The data model in mmCIF is more general. It comprises the data items **_database_2.database_id**, which identifies the database, and **_database_2.database_code**, which is the code assigned by the database to the entry. Thus a new database can be referred to without needing to add an additional data item to the dictionary. If a structure has been deposited in more than one database, the values of **_database_2.database_id** and **_database_2.database_code** can be looped.

The institutions and individual databases recognized in the DATABASE_2 category in the current version of the mmCIF dictionary are CAS (*Chemical Abstracts* Service), CSD (Cambridge Structural Database), ICSD (Inorganic Crystal Structure Database), MDF (Metals Data File), NDB (Nucleic Acid Database), NBS (the Crystal Data database of the National Institute of Standards and Technology, formerly the National Bureau of Standards), PDB (Protein Data Bank), PDF (Powder Diffraction File), RCSB (Research Collaboratory for Structural Bioinformatics) and EBI (European Bioinformatics Institute). It is intended that new databases will be added to this list on an ongoing basis; the purpose of specifying a list of possible databases in the dictionary is to ensure that each database is referenced consistently.

3.6.8.3.2. *Compatibility with PDB format files*

Data items in these categories are as follows:

(*a*) DATABASE_PDB_REV

- **_database_PDB_rev.num**
  _database_PDB_rev.author_name
  _database_PDB_rev.date
  _database_PDB_rev.date_original
  _database_PDB_rev.mod_type
  _database_PDB_rev.replaced_by
  _database_PDB_rev.replaces
  _database_PDB_rev.status

(*b*) DATABASE_PDB_REV_RECORD

- **_database_PDB_rev_record.rev_num**
  → **_database_PDB_rev.num**
- **_database_PDB_rev_record.type**
  _database_PDB_rev_record.details

(*c*) DATABASE_PDB_MATRIX

- **_database_PDB_matrix.entry_id**
  → **_entry.id**
  _database_PDB_matrix.origx[1][1]
  _database_PDB_matrix.origx[1][2]
  _database_PDB_matrix.origx[1][3]
  _database_PDB_matrix.origx[2][1]
  _database_PDB_matrix.origx[2][2]
  _database_PDB_matrix.origx[2][3]
  _database_PDB_matrix.origx[3][1]
  _database_PDB_matrix.origx[3][2]
  _database_PDB_matrix.origx[3][3]
  _database_PDB_matrix.origx_vector[1]
  _database_PDB_matrix.origx_vector[2]
  _database_PDB_matrix.origx_vector[3]
  _database_PDB_matrix.scale[1][1]
  _database_PDB_matrix.scale[1][2]
  _database_PDB_matrix.scale[1][3]
  _database_PDB_matrix.scale[2][1]
  _database_PDB_matrix.scale[2][2]
  _database_PDB_matrix.scale[2][3]
  _database_PDB_matrix.scale[3][1]
  _database_PDB_matrix.scale[3][2]
  _database_PDB_matrix.scale[3][3]
  _database_PDB_matrix.scale_vector[1]
  _database_PDB_matrix.scale_vector[2]
  _database_PDB_matrix.scale_vector[3]

(*d*) DATABASE_PDB_TVECT

- **_database_PDB_tvect.id**
  _database_PDB_tvect.details
  _database_PDB_tvect.vector[1]
  _database_PDB_tvect.vector[2]
  _database_PDB_tvect.vector[3]

(*e*) DATABASE_PDB_CAVEAT

- **_database_PDB_caveat.id**
  _database_PDB_caveat.text

(*f*) DATABASE_PDB_REMARK

- **_database_PDB_remark.id**
  _database_PDB_remark.text

The bullet (•) *indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow* (→) *is a reference to a parent data item.*

A major goal of the design of the mmCIF data model was that a file could be transformed from Protein Data Bank (PDB) format to mmCIF format and back again without loss of information. This required the creation of mmCIF data items whose sole purpose is to capture PDB-specific records that do not map onto mmCIF data items. These records would never be created for a *de novo* mmCIF. This family of categories also belongs to the PDB category group (see Section 3.6.9.3).

The items in the categories DATABASE_PDB_MATRIX and DATABASE_PDB_TVECT are derived from the elements of transformation matrices and vectors used by the Protein Data Bank. The items in the categories DATABASE_PDB_REV and DATABASE_PDB_REV_RECORD record details about the revision history of the data block as archived by the Protein Data Bank.

The items in the DATABASE_PDB_CAVEAT category record comments about the data block flagged as 'CAVEATS' by the Protein Data Bank at the time the original PDB archive file was created. A PDB CAVEAT record indicates that the entry contains severe errors. In PDB format, extended comments were stored as a sequence of fixed-length (80-character) format records, columns 9 and 10 being reserved for continuation sequence numbering. The mmCIF representation retains each record as a separate data value and does not attempt to merge continuation records to provide more readable running text. Hence the PDB CAVEAT entry

```
CAVEAT     1ABC    THE CRYSTAL TRANSFORMATION IS WRONG
CAVEAT   2 1ABC    BUT IS UNCORRECTABLE AT THIS TIME
```

would be represented in mmCIF as

```
loop_
  _database_PDB_caveat.id
     _database_PDB_caveat.text
    1
; THE CRYSTAL TRANSFORMATION IS WRONG
;
    2
; BUT IS UNCORRECTABLE AT THIS TIME
;
```

The PDB format used 'REMARK' records to store information relating to several aspects of the structure in free or loosely structured text. In some cases, the conventions used for individual types of REMARK record allow structured data to be extracted automatically and translated to specific mmCIF data items. Where this is not possible, the DATABASE_PDB_REMARK category may be used to retain the information that appeared in these parts of PDB format files. Unlike the CAVEAT records, it is possible to collect together several REMARK records sharing a common numbering into a single free-text field. For example, PDB practice has been to repeat the contents of CAVEAT records (see above) as records of type 'REMARK 5'. While each separate CAVEAT record is converted to a separate mmCIF data value, the complete text of a REMARK 5 record may be gathered into a single mmCIF data value. Hence the CAVEAT example above would also appear in a PDB file as part of a 'REMARK 5' as

```
REMARK   5 THE CRYSTAL TRANSFORMATION IS WRONG
REMARK   5 BUT IS UNCORRECTABLE AT THIS TIME
```

and would appear in an mmCIF as

```
loop_
  _database_PDB_remark.id
     _database_PDB_remark.text
    5
; THE CRYSTAL TRANSFORMATION IS WRONG
  BUT IS UNCORRECTABLE AT THIS TIME
;
```

Note that by convention the value of `_database_PDB_remark.id` matches the class of the REMARK record in the PDB file.

3.6.8.4. Article publication

Categories used during the publication of an article are as follows:

IUCR group

Journal housekeeping and reference entries (§3.6.8.4.1)

- JOURNAL
- JOURNAL_INDEX

Contents of a publication (§3.6.8.4.2)

- PUBL
- PUBL_AUTHOR
- PUBL_BODY
- PUBL_MANUSCRIPT_INCL

These categories cover both the metadata for the article (information about the article) and the text of the article itself.

3.6.8.4.1. *Journal housekeeping and citation entries*

Data items in these categories are as follows:

(*a*) JOURNAL

- **`_journal.entry_id`**
  → **`_entry.id`**

`_journal.coden_ASTM`
`_journal.coden_Cambridge`
`_journal.coeditor_address`
`_journal.coeditor_code`
`_journal.coeditor_email`
`_journal.coeditor_fax`
`_journal.coeditor_name`
`_journal.coeditor_notes`
`_journal.coeditor_phone`
`_journal.data_validation_number`
`_journal.date_accepted`
`_journal.date_from_coeditor`
`_journal.date_to_coeditor`
`_journal.date_printers_final`
`_journal.date_printers_first`
`_journal.date_proofs_in`
`_journal.date_proofs_out`
`_journal.date_recd_copyright`
`_journal.date_recd_electronic`
`_journal.date_recd_hard_copy`
`_journal.issue`
`_journal.language`
`_journal.name_full`
`_journal.page_first`
`_journal.page_last`
`_journal.paper_category`
`_journal.suppl_publ_number`
`_journal.suppl_publ_pages`
`_journal.techeditor_address`
`_journal.techeditor_code`
`_journal.techeditor_email`
`_journal.techeditor_fax`
`_journal.techeditor_name`
`_journal.techeditor_notes`
`_journal.techeditor_phone`
`_journal.volume`
`_journal.year`

(*b*) JOURNAL_INDEX

`_journal_index.subterm`
`_journal_index.term`
`_journal_index.type`

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_).

In mmCIF, the families of categories used to contain the text of an article for publication and to record information about the handling and processing of the article by a publisher are assigned to the IUCR category group. The name arose from the fact that CIF is sponsored by the International Union of Crystallography and several of the journals of the IUCr can handle articles submitted for publication in CIF format. However, these data items may be

freely used by other publishers who wish to handle articles submitted in CIF format. The JOURNAL and JOURNAL_INDEX categories are used in the same way in the core CIF and mmCIF dictionaries, and Section 3.2.5.4 can be consulted for details.

3.6.8.4.2. *Contents of a publication*

Data items in these categories are as follows:

(*a*) PUBL

• **_publ.entry_id**
  → **_entry.id**
  _publ.contact_author
  _publ.contact_author_address
  _publ.contact_author_email
  _publ.contact_author_fax
  _publ.contact_author_name
  _publ.contact_author_phone
  _publ.contact_letter
  _publ.manuscript_creation
  _publ.manuscript_processed
  _publ.manuscript_text
  _publ.requested_category
  _publ.requested_coeditor_name
  _publ.requested_journal
  _publ.section_abstract
  _publ.section_acknowledgements
  _publ.section_comment
  _publ.section_discussion
  _publ.section_experimental
  _publ.section_exptl_prep
  _publ.section_exptl_refinement
  _publ.section_exptl_solution
  _publ.section_figure_captions
  _publ.section_introduction
  _publ.section_references
  _publ.section_synopsis
  _publ.section_table_legends
  _publ.section_title
  _publ.section_title_footnote

(*b*) PUBL_AUTHOR
  _publ_author.address
  _publ_author.email
  _publ_author.footnote
  _publ_author.id_iucr
  _publ_author.name

(*c*) PUBL_BODY
  _publ_body.contents
  _publ_body.element
  _publ_body.format
  _publ_body.label
  _publ_body.title

(*d*) PUBL_MANUSCRIPT_INCL

• **_publ_manuscript_incl.entry_id**
  → **_entry.id**
  _publ_manuscript_incl.extra_defn
  _publ_manuscript_incl.extra_info
  _publ_manuscript_incl.extra_item

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_).

The categories PUBL, PUBL_AUTHOR, PUBL_BODY and PUBL_MANUSCRIPT_INCL are also members of the IUCR group in the mmCIF dictionary. They are used in the same way in the core CIF and mmCIF dictionaries, and Section 3.2.5.5 can be consulted for details.

3.6.9. File metadata

As in the core CIF dictionary, information about the source and the revision history of an mmCIF may be given in the AUDIT group of categories: AUDIT, AUDIT_AUTHOR, AUDIT_CONTACT_AUTHOR and AUDIT_CONFORM (Section 3.6.9.1). However, the mmCIF dictionary differs from the core CIF dictionary in the way it expresses relationships between data blocks: instead of the core AUDIT_LINK category, mmCIF has two categories, ENTRY and ENTRY_LINK, that essentially fulfil the same role but are classified in a distinct category group (Section 3.6.9.2).

3.6.9.1. History of a data block

The categories describing the history of a data block are as follows:

AUDIT group
  AUDIT
  AUDIT_AUTHOR
  AUDIT_CONFORM
  AUDIT_CONTACT_AUTHOR

Data items in these categories are as follows:

(*a*) AUDIT

• **_audit.revision_id**
  _audit.creation_date
  _audit.creation_method
  _audit.update_record

(*b*) AUDIT_AUTHOR

• ***_audit_author.name***
  _audit_author.address

(*c*) AUDIT_CONFORM

• ***_audit_conform.dict_name***
• ***_audit_conform.dict_version***
  _audit_conform.dict_location

(*d*) AUDIT_CONTACT_AUTHOR

• ***_audit_contact_author.name***
  _audit_contact_author.address
  _audit_contact_author.email
  _audit_contact_author.fax
  _audit_contact_author.phone

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_).

The data items in these categories are used in the same way in the mmCIF dictionary as in the core CIF dictionary (see Section 3.2.6). The data item **_audit.revision_id** has been added to the AUDIT category to provide the formal category key required by the DDL2 data model. The core data item **_audit_block_code** has been replaced by **_entry.id** (see Section 3.6.9.2).

3.6.9.2. Links between data blocks

The categories describing links between data blocks are as follows:

ENTRY group
  ENTRY
  ENTRY_LINK

AUDIT group
  AUDIT_LINK

Data items in these categories are as follows:

(*a*) ENTRY

• **_entry.id**

(*b*) ENTRY_LINK

• **_entry_link.entry_id**
  → **_entry.id**
• **_entry_link.id**
  _entry_link.details

(*c*) AUDIT_LINK
- **_audit_link.block_code**
- **_audit_link.block_description**

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item. Items in italics have aliases in the core CIF dictionary formed by changing the full stop (.) to an underscore (_).

The sole data item in the category ENTRY, **_entry.id**, is a label that identifies the current data block. This label is used as the formal key in several categories that record information that is relevant to the entire data block (*e.g.* **_cell.entry_id**, **_geom.entry_id**), so care should be taken to select a label that is informative and unique.

Data items in the ENTRY_LINK category record the relationships between the current data block and other data blocks within the current file which may be referenced in the current data block. Since there are no formal constraints on the value of **_entry.id** assigned to each data block, authors must take care to ensure that an mmCIF comprised of several distinct data blocks uses a different value for **_entry.id** in each block.

As mentioned in the introductory paragraph of Section 3.6.9, the ENTRY_LINK category is used in mmCIF applications instead of the core category AUDIT_LINK. The latter is retained formally in the mmCIF dictionary for strict compatibility with the core dictionary, and the data items in this category, **_audit_link.blockcode** and **_audit_link.block_description**, are aliased to corresponding core data names (see Section 3.2.6.1). Their use is not recommended in mmCIF applications.

3.6.9.3. Other category classifications

The following categories, already described elsewhere in this chapter, are included in other formal category groups:

Compliance with earlier dictionaries

COMPLIANCE group
- DATABASE

Compatibility with PDB format files

PDB group
- DATABASE_PDB_CAVEAT
- DATABASE_PDB_MATRIX
- DATABASE_PDB_REMARK
- DATABASE_PDB_REV
- DATABASE_PDB_REV_RECORD
- DATABASE_PDB_TVECT

The COMPLIANCE group includes categories that appear in the mmCIF dictionary for the sole purpose of ensuring compliance with earlier dictionaries. They are not intended for use in the creation of new mmCIFs. As was discussed in Section 3.6.8.3, the DATABASE category of the core CIF is replaced in mmCIF by the more structured DATABASE_2 category. Thus the core CIF DATABASE category appears in the mmCIF COMPLIANCE group. At the time of writing (2005), DATABASE is the only category in the COMPLIANCE group.

The PDB group includes a number of categories that record unstructured information imported from various records in Protein Data Bank (PDB) format files. These categories are also part of the DATABASE group and were discussed in Section 3.6.8.3.2.

Appendix 3.6.1
Category structure of the mmCIF dictionary

Table A3.6.1.1 provides an overview of the structure of the mmCIF dictionary by category group and member categories.

Appendix 3.6.2
The Protein Data Bank exchange data dictionary

BY J. D. WESTBROOK, K. HENRICK, E. L. ULRICH AND H. M. BERMAN

In developing a data-management infrastructure, the Protein Data Bank (PDB; Berman *et al.*, 2000) has chosen the mmCIF dictionary technology for describing the data that it collects and disseminates. To accommodate the growth in the PDB's activities, data collection, processing and annotation now occur at three sites worldwide: the Research Collaboratory for Structural Bioinformatics (RCSB/PDB), the Macromolecular Structural Database (MSD) at the European Bioinformatics Institute (EBI) and the Protein Data Bank Japan (PDBj) at Osaka. Together these facilities form the Worldwide PDB (wwPDB) (Berman *et al.*, 2003). In order to maintain the fidelity of the single archive of three-dimensional macromolecular structure, a precise content description is required to support the accurate exchange of data among the different sites and the exchange of information between different file formats.

A key strength of the mmCIF technology is the extensibility afforded by a framework based on a software-accessible data dictionary. The PDB has exploited this functionality by using the mmCIF dictionary as a foundation and supplementing it with extensions in order to describe all aspects of data processing and database operations.

These extensions include content required to support reversible format translation, noncrystallographic structure determination methods and the details of protein production. They also support recommendations by the International Union of Crystallography (IUCr) and the International Structural Genomics Organization (ISGO) as to which data should be deposited. In the following sections, the extensions to the mmCIF data dictionary developed by the PDB (http://mmcif.pdb.org/) are described.

A3.6.2.1. Data exchange and format translation

The majority of crystallographic and structural concepts embodied in the PDB are already well described in the mmCIF data dictionary. However, while there is a conceptual description of most crystallographic information in PDB-format files within the mmCIF dictionary, the precise representation of this information can differ subtly. To guarantee accurate data exchange and to facilitate reversible format translation between PDB and mmCIF formats, all such differences in representation must be resolved.

To accommodate content and semantic differences between formats, extensions to the dictionary have been created. These extensions take one of two forms: the addition of new definitions to existing categories or the creation of new categories. Where possible, extensions are added to existing categories. This is done when the new definition supplements the content of the category without changing the category definition or its fundamental organization. However, if a new definition cannot be added to an existing category, a new category is created to hold the extension. All new data items and categories include the prefix **pdbx_** in their names.

For example, the level of detail in the PDB description of the biological source exceeds the description provided by mmCIF. In this case, dictionary extensions have been added to the existing categories ENTITY_SRC_NAT and ENTITY_SRC_GEN (where 'nat' and 'gen' stand for naturally occurring and genetically engineered, respectively). The PDB description of atomic coordinates includes two items that are not described in mmCIF: the insertion code

Table A3.6.1.1. *Categories in the mmCIF dictionary*

Numbers in parentheses refer to the section of this chapter in which each category is described in detail.

ATOM group (§3.6.7.1)
- ATOM_SITE (§3.6.7.1.1(*a*))
- ATOM_SITE_ANISOTROP (§3.6.7.1.1(*b*))
- ATOM_SITES (§3.6.7.1.2(*a*))
- ATOM_SITES_ALT (§3.6.7.1.4(*a*))
- ATOM_SITES_ALT_ENS (§3.6.7.1.4(*b*))
- ATOM_SITES_ALT_GEN (§3.6.7.1.4(*c*))
- ATOM_SITES_FOOTNOTE (§3.6.7.1.2(*b*))
- ATOM_TYPE (§3.6.7.1.3)

AUDIT group (§3.6.9.1)
- AUDIT (§3.6.9.1(*a*))
- AUDIT_AUTHOR (§3.6.9.1(*b*))
- AUDIT_CONFORM (§3.6.9.1(*c*))
- AUDIT_CONTACT_AUTHOR (§3.6.9.1(*d*))
- AUDIT_LINK (§3.6.9.2(*c*))

CELL group (§3.6.5.1)
- CELL (§3.6.5.1(*a*))
- CELL_MEASUREMENT (§3.6.5.1(*b*))
- CELL_MEASUREMENT_REFLN (§3.6.5.1(*c*))

CHEM_COMP group (§3.6.7.2.2)
- CHEM_COMP (§3.6.7.2.2(*a*))
- CHEM_COMP_ANGLE (§3.6.7.2.2(*b*))
- CHEM_COMP_ATOM (§3.6.7.2.2(*c*))
- CHEM_COMP_BOND (§3.6.7.2.2(*d*))
- CHEM_COMP_CHIR (§3.6.7.2.2(*e*))
- CHEM_COMP_CHIR_ATOM (§3.6.7.2.2(*f*))
- *CHEM_COMP_LINK* (*see CHEM_LINK group*)
- CHEM_COMP_PLANE (§3.6.7.2.2(*h*))
- CHEM_COMP_PLANE_ATOM (§3.6.7.2.2(*i*))
- CHEM_COMP_TOR (§3.6.7.2.2(*j*))
- CHEM_COMP_TOR_VALUE (§3.6.7.2.2(*k*))

CHEM_LINK group (§3.6.7.2.3)
- CHEM_COMP_LINK (§3.6.7.2.2(*g*))
- CHEM_LINK (§3.6.7.2.3(*a*))
- CHEM_LINK_ANGLE (§3.6.7.2.3(*b*))
- CHEM_LINK_BOND (§3.6.7.2.3(*c*))
- CHEM_LINK_CHIR (§3.6.7.2.3(*d*))
- CHEM_LINK_CHIR_ATOM (§3.6.7.2.3(*e*))
- CHEM_LINK_PLANE (§3.6.7.2.3(*f*))
- CHEM_LINK_PLANE_ATOM (§3.6.7.2.3(*g*))
- CHEM_LINK_TOR (§3.6.7.2.3(*h*))
- CHEM_LINK_TOR_VALUE (§3.6.7.2.3(*i*))
- ENTITY_LINK (§3.6.7.2.3(*j*))

CHEMICAL group (§3.6.7.2)
- CHEMICAL (§3.6.7.2.1(*a*))
- CHEMICAL_CONN_ATOM (§3.6.7.2.1(*b*))
- CHEMICAL_CONN_BOND (§3.6.7.2.1(*c*))
- CHEMICAL_FORMULA (§3.6.7.2.1(*d*))

CITATION group (§3.6.8.1)
- CITATION (§3.6.8.1(*a*))
- CITATION_AUTHOR (§3.6.8.1(*b*))
- CITATION_EDITOR (§3.6.8.1(*c*))

COMPUTING group (§3.6.8.2)
- COMPUTING (§3.6.8.2(*a*))
- SOFTWARE (§3.6.8.2(*b*))

DATABASE group (§§3.6.8.3, 3.6.9.3)
- DATABASE (§3.6.8.3.1(*a*))
- DATABASE_2 (§3.6.8.3.1(*b*))
- *The following also belong to the PDB group*
- DATABASE_PDB_CAVEAT (§3.6.8.3.2(*e*))
- DATABASE_PDB_MATRIX (§3.6.8.3.2(*c*))
- DATABASE_PDB_REMARK (§3.6.8.3.2(*f*))
- DATABASE_PDB_REV (§3.6.8.3.2(*a*))
- DATABASE_PDB_REV_RECORD (§3.6.8.3.2(*b*))
- DATABASE_PDB_TVECT (§3.6.8.3.2(*d*))

DIFFRN group (§3.6.5.2)
- DIFFRN (§3.6.5.2(*a*))
- DIFFRN_ATTENUATOR (§3.6.5.2(*b*))
- DIFFRN_DETECTOR (§3.6.5.2(*c*))
- DIFFRN_MEASUREMENT (§3.6.5.2(*d*))
- DIFFRN_ORIENT_MATRIX (§3.6.5.2(*e*))
- DIFFRN_ORIENT_REFLN (§3.6.5.2(*f*))
- DIFFRN_RADIATION (§3.6.5.2(*g*))
- DIFFRN_RADIATION_WAVELENGTH (§3.6.5.2(*h*))
- DIFFRN_REFLN (§3.6.5.2(*i*))
- DIFFRN_REFLNS (§3.6.5.2(*j*))
- DIFFRN_REFLNS_CLASS (§3.6.5.2(*k*))
- DIFFRN_SCALE_GROUP (§3.6.5.2(*l*))
- DIFFRN_SOURCE (§3.6.5.2(*m*))
- DIFFRN_STANDARD_REFLN (§3.6.5.2(*n*))
- DIFFRN_STANDARDS (§3.6.5.2(*o*))

ENTITY group (§3.6.7.3)
- ENTITY (§3.6.7.3.1(*a*))
- ENTITY_KEYWORDS (§3.6.7.3.1(*b*))
- *ENTITY_LINK* (*see CHEM_LINK group*)
- ENTITY_NAME_COM (§3.6.7.3.1(*c*))
- ENTITY_NAME_SYS (§3.6.7.3.1(*d*))
- ENTITY_POLY (§3.6.7.3.2(*a*))
- ENTITY_POLY_SEQ (§3.6.7.3.2(*b*))
- ENTITY_SRC_GEN (§3.6.7.3.1(*e*))
- ENTITY_SRC_NAT (§3.6.7.3.1(*f*))

ENTRY group (§3.6.9.2)
- ENTRY (§3.6.9.2(*a*))
- ENTRY_LINK (§3.6.9.2(*b*))

EXPTL group (§3.6.5.3)
- EXPTL (§3.6.5.3.1(*a*))
- EXPTL_CRYSTAL (§3.6.5.3.1(*b*))
- EXPTL_CRYSTAL_FACE (§3.6.5.3.1(*c*))
- EXPTL_CRYSTAL_GROW (§3.6.5.3.2(*a*))
- EXPTL_CRYSTAL_GROW_COMP (§3.6.5.3.2(*b*))

GEOM group (§3.6.7.4)
- GEOM (§3.6.7.4(*a*))
- GEOM_ANGLE (§3.6.7.4(*b*))
- GEOM_BOND (§3.6.7.4(*c*))
- GEOM_CONTACT (§3.6.7.4(*d*))
- GEOM_HBOND (§3.6.7.4(*e*))
- GEOM_TORSION (§3.6.7.4(*f*))

IUCR group (§3.6.8.4)
- JOURNAL (§3.6.8.4.1(*a*))
- JOURNAL_INDEX (§3.6.8.4.1(*b*))
- PUBL (§3.6.8.4.2(*a*))
- PUBL_AUTHOR (§3.6.8.4.2(*b*))
- PUBL_BODY (§3.6.8.4.2(*c*))
- PUBL_MANUSCRIPT_INCL (§3.6.8.4.2(*d*))

PHASING group (§3.6.6.1)
- PHASING (§3.6.6.1.1)
- PHASING_AVERAGING (§3.6.6.1.2)
- PHASING_ISOMORPHOUS (§3.6.6.1.3)
- PHASING_MAD (§3.6.6.1.4(*a*))
- PHASING_MAD_CLUST (§3.6.6.1.4(*b*))
- PHASING_MAD_EXPT (§3.6.6.1.4(*c*))
- PHASING_MAD_RATIO (§3.6.6.1.4(*d*))
- PHASING_MAD_SET (§3.6.6.1.4(*e*))
- PHASING_MIR (§3.6.6.1.5(*a*))
- PHASING_MIR_DER (§3.6.6.1.5(*c*))
- PHASING_MIR_DER_REFLN (§3.6.6.1.5(*d*))
- PHASING_MIR_DER_SHELL (§3.6.6.1.5(*e*))
- PHASING_MIR_DER_SITE (§3.6.6.1.5(*f*))
- PHASING_MIR_SHELL (§3.6.6.1.5(*b*))
- PHASING_SET (§3.6.6.1.6(*a*))
- PHASING_SET_REFLN (§3.6.6.1.6(*b*))
- PUBL (*see IUCR group*)
- PUBL_AUTHOR (*see IUCR group*)
- PUBL_BODY (*see IUCR group*)
- PUBL_MANUSCRIPT_INCL (*see IUCR group*)

REFINE group (§3.6.6.2)
- REFINE (§3.6.6.2.1(*a*))
- REFINE_ANALYZE (§3.6.6.2.2)
- REFINE_B_ISO (§3.6.6.2.4(*a*))
- REFINE_FUNCT_MINIMIZED (§3.6.6.2.1(*b*))
- REFINE_HIST (§3.6.6.2.5)
- REFINE_LS_RESTR (§3.6.6.2.3(*a*))
- REFINE_LS_RESTR_NCS (§3.6.6.2.3(*b*))
- REFINE_LS_CLASS (§3.6.2.2.3(*e*))
- REFINE_LS_RESTR_TYPE (§3.6.6.2.3(*c*))
- REFINE_LS_SHELL (§3.6.6.2.3(*d*))
- REFINE_OCCUPANCY (§3.6.6.2.4(*b*))

REFLN group (§3.6.6.3)
- REFLN (§3.6.6.3.1(*a*))
- REFLN_SYS_ABS (§3.6.6.3.1(*b*))
- REFLNS (§3.6.6.3.2(*a*))
- REFLNS_CLASS (§3.6.6.3.2(*d*))
- REFLNS_SCALE (§3.6.6.3.2(*b*))
- REFLNS_SHELL (§3.6.6.3.2(*c*))

- *SOFTWARE* (*see COMPUTING group*)
- *SPACE_GROUP* (*see SYMMETRY group*)
- *SPACE_GROUP_SYMOP* (*see SYMMETRY group*)

STRUCT group (§3.6.7.5)
- STRUCT (§3.6.7.5.1(*a*))
- STRUCT_ASYM (§3.6.7.5.1(*b*))
- STRUCT_BIOL (§3.6.7.5.1(*c*))
- STRUCT_BIOL_GEN (§3.6.7.5.1(*d*))
- STRUCT_BIOL_KEYWORDS (§3.6.7.5.1(*e*))
- STRUCT_BIOL_VIEW (§3.6.7.5.1(*f*))
- STRUCT_CONF (§3.6.7.5.2(*b*))
- STRUCT_CONF_TYPE (§3.6.7.5.2(*a*))
- STRUCT_CONN (§3.6.7.5.3(*b*))
- STRUCT_CONN_TYPE (§3.6.7.5.3(*a*))
- STRUCT_KEYWORDS (§3.6.7.5.1(*g*))
- STRUCT_MON_DETAILS (§3.6.7.5.4(*a*))
- STRUCT_MON_NUCL (§3.6.7.5.4(*b*))
- STRUCT_MON_PROT (§3.6.7.5.4(*c*))
- STRUCT_MON_PROT_CIS (§3.6.7.5.4(*d*))
- STRUCT_NCS_DOM (§3.6.7.5.5(*c*))
- STRUCT_NCS_DOM_LIM (§3.6.7.5.5(*d*))
- STRUCT_NCS_ENS (§3.6.7.5.5(*a*))
- STRUCT_NCS_ENS_GEN (§3.6.7.5.5(*b*))
- STRUCT_NCS_OPER (§3.6.7.5.5(*e*))
- STRUCT_REF (§3.6.7.5.6(*a*))
- STRUCT_REF_SEQ (§3.6.7.5.6(*b*))
- STRUCT_REF_SEQ_DIF (§3.6.7.5.6(*c*))
- STRUCT_SHEET (§3.6.7.5.7(*a*))
- STRUCT_SHEET_HBOND (§3.6.7.5.7(*e*))
- STRUCT_SHEET_ORDER (§3.6.7.5.7(*d*))
- STRUCT_SHEET_RANGE (§3.6.7.5.7(*c*))
- STRUCT_SHEET_TOPOLOGY (§3.6.7.5.7(*b*))
- STRUCT_SITE (§3.6.7.5.8(*a*))
- STRUCT_SITE_GEN (§3.6.7.5.8(*c*))
- STRUCT_SITE_KEYWORDS (§3.6.7.5.8(*b*))
- STRUCT_SITE_VIEW (§3.6.7.5.8(*d*))

SYMMETRY group (§3.6.7.6)
- SPACE_GROUP (§3.6.7.6(*c*))
- SPACE_GROUP_SYMOP (§3.6.7.6(*d*))
- SYMMETRY (§3.6.7.6(*a*))
- SYMMETRY_EQUIV (§3.6.7.6(*b*))

VALENCE group (§3.6.7.7)
- VALENCE_PARAM group (§3.6.7.7(*a*))
- VALENCE_REF group (§3.6.7.7(*b*))

and the model number. These have been added to the mmCIF category ATOM_SITE (as **`_atom_site.pdbx_PDB_ins_code`** and **`_atom_site.pdbx_PDB_model_num`**) and to all related categories that include atom nomenclature.

The convention for defining the hydrogen bonding in β-sheets differs between the PDB and mmCIF representations. Because the PDB model is fundamentally different from that found in mmCIF, a new category was created to hold the PDB data: PDBX_STRUCT_SHEET_HBOND. The correspondence between the PDB and mmCIF formats is tabulated at http://deposit.pdb.org/mmCIF/dictionaries/pdb-correspondence/pdb2mmcif.html.

A3.6.2.2. Extensions for structural genomics

An International Task Force on Deposition, Archiving, and Curation of Primary Information for Structural Genomics was formed under the auspices of the International Structural Genomics Organization (ISGO) in 2001 (Berman, 2001) and was asked to develop specifications for data from structural genomics projects to be deposited with the PDB. The recommendations from this working group are summarized at http://deposit.pdb.org/mmcif/sg-data/xstal.html and http://deposit.pdb.org/mmcif/sg-data/nmr.html. For data from crystallography-based projects, the content extensions are largely focused on a more detailed description of phasing, tracing and density modification. All of the ISGO recommendations have been incorporated into the PDB exchange dictionary.

A3.6.2.3. Noncrystallographic methods

The IUCr-sponsored development of data dictionaries has been focused exclusively on crystallographic methods. As the repository for all three-dimensional macromolecular structure data, the PDB accepts structures determined using noncrystallographic techniques such as NMR and cryo-electron microscopy. The description of noncrystallographic methods is beyond the remit of the IUCr, so the PDB has worked with the NMR and cryo-electron microscopy communities to develop data dictionaries that describe these techniques within the mmCIF framework.

A3.6.2.3.1. *NMR*

The PDB exchange dictionary includes a description of NMR sample preparation, structure solution methodology, refinement and refinement metrics. These extensions were developed in collaboration with the BioMagResBank (BMRB; Ulrich *et al.*, 1989). The BMRB is the archive for experimental NMR data for biological macromolecules and has played an active role in the development of the mmCIF data dictionary. In selecting a format for archiving NMR data, the BMRB opted to use the STAR syntax (Hall, 1991) rather than the more restrictive CIF syntax. Despite this difference in syntax, the conceptual representation of macromolecular structure in the NMR dictionary (NMRStar) has remained semantically very close to the mmCIF representation. This has facilitated the exchange of data and dictionaries between the BMRB and the PDB, the sharing of software tools, and the development of a common platform for depositing data.

A3.6.2.3.2. *Cryo-electron microscopy*

Cryo-electron microscopy (as a technique for the determination of the structure of large molecular assemblies) is also described in the PDB exchange dictionary. The data extensions for cryo-electron microscopy include a description of the sample preparation, raw volume data (Henrick *et al.*, 2003), structure solution and refinement. These extensions have a prefix of `em_` (http://mmcif.pdb.org/dictionaries/mmcif_iims.dic/Index/).

A3.6.2.3.3. *Protein production*

The International Task Force on Deposition, Archiving, and Curation of Primary Information for Structural Genomics (Section A3.6.2.2) has also provided recommendations for the deposition of information about protein production. These recommendations are summarized at http://deposit.pdb.org/mmcif/sg-data/protprod.html. These data extensions have been used as the foundation for the Protein Expression Purification and Crystallization database (PEPCdb, http://pepcdb.pdb.org/) and for the protein production process model developed to support the Structural Proteomics in Europe initiative (SPINE; http://www.spineurope.org/).

A3.6.2.4. Supporting software

The RCSB/PDB has developed a set of software tools which support the PDB exchange dictionary framework (Chapter 5.5). These include *PDB_EXTRACT*, a tool to extract data from the output files of structure determination applications; *ADIT*, a web-based editor for data files based on the PDB exchange dictionary; and *CIFTr*, a translator from mmCIF to PDB format. These applications and other supporting utilities can be downloaded from http://sw-tools.pdb.org/.

The development of the mmCIF dictionary and DDL2 has been an enormous task, and any list of contributors to the effort will certainly be incomplete. Still, we must try. We have so appreciated the people that have taken the time to think carefully and constructively about all of this, and we would like to recognize their efforts. We begin by recognizing Syd Hall, David Brown and Frank Allen, who began the entire CIF effort and who recruited us to do the extensions for macromolecular structure.

Chapter 1.1 describes the formation of the original mmCIF working group, chaired by Paula Fitzgerald and including Enrique Abola, Helen Berman, Phil Bourne, Eleanor Dodson, Art Olson, Wolfgang Steigemann, Lynn Ten Eyck and Keith Watenpaugh. However, the number of people who contributed to the original design of the mmCIF data structure is much larger. We would like to thank Steve Bryant, Vivian Stojanoff, Jean Richelle, Eldon Ulrich and Brian Toby.

There are also the people who realized the shortcomings of the original DDL and worked hard to convince us that a more rigorous underpinning for the dictionary would be needed. Among them are Michael Scharf, Peter Grey, Peter Murray-Rust, Dave Stampf and Jan Zelinka.

Writing the dictionary and developing the new DDL were just the starting points for evaluation and critique, and this effort has been greatly aided by the input from COMCIFS, the IUCr committee with oversight over this process (David Brown, Chair). But the real process of review, after the dictionary was released to the public for comment in August 1995, has involved a much larger number of people. We cannot say enough about the valuable input we have received from Frances Bernstein, Herbert Bernstein, Dale Tronrud and Peter Keller.

Our efforts have been greatly enabled by the staff of the Nucleic Acid Database at Rutgers University, who have dealt with many of the technical issues of the implementation of mmCIF with real data. So we would also like to thank Anke Gelbin, Shu-Hsin Hsieh and Christine Zardecki.

Without the three CIF workshops described in Chapter 1.1, this effort would never have taken the shape and focus it now has, and we are eternally gratefully to Eleanor Dodson (York), Phil Bourne (Tarrytown) and Shoshana Wodak (Brussels), who organized the workshops, and also to Helen Berman and John Westbrook for hosting the subsequent workshop at Rutgers following the publication of the mmCIF dictionary. We thank the European Science Foundation (ESF), the European Union (EU), the National Science Foundation (NSF) and the US Department of Energy (DOE), who provided the funding.

The RCSB/PDB is operated by Rutgers, The State University of New Jersey; the San Diego Supercomputer Center at the University of California, San Diego; and the Center for Advanced Research in Biotechnology/UMBI/NIST. RCSB/PDB is supported by funds

from the National Science Foundation (NSF), the National Institute of General Medical Sciences (NIGMS), the Office of Science, Department of Energy (DOE), the National Library of Medicine (NLM), the National Cancer Institute (NCI), the National Center for Research Resources (NCRR), the National Institute of Biomedical Imaging and Bioengineering (NIBIB) and the National Institute of Neurological Disorders and Stroke (NINDS).

References

Altona, C. & Sundaralingam, M. (1972). *Conformational analysis of the sugar ring in nucleosides and nucleotides. New description using the concept of pseudorotation. J. Am. Chem. Soc.* **94**, 8205–8212.

Berman, H. M. (2001). Chair. *Report of the task force on the deposition, archiving, and curation of the primary information.* Task Force Reports from the Second International Structural Genomics Meeting, Airlie, Virginia, USA. http://www.nigms.nih.gov/news/reports/airlie_tasks.html.

Berman, H. M., Henrick, K. & Nakamura, H. (2003). *Announcing the worldwide Protein Data Bank. Nature Struct. Biol.* **10**, 980.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *The Protein Data Bank. Nucleic Acids Res.* **28**, 235–242.

Bourne, P., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. & Fitzgerald, P. M. D. (1997). *Macromolecular Crystallographic Information File. Methods Enzymol.* **277**, 571–590.

Brändén C.-I. & Jones, T. A. (1990). *Between objectivity and subjectivity. Nature (London)*, **343**, 687–689.

Brünger, A. T. (1997). *Free R value: cross-validation in crystallography. Methods Enzymol.* **277**, 366–396.

Cruickshank, D. W. J. (1999). *Remarks about protein structure precision. Acta Cryst.* D**55**, 583–601.

Driessen, H., Haneef, M. I. J., Harris, G. W., Howlin, B., Khan, G. & Moss, D. S. (1989). *RESTRAIN: restrained structure-factor least-squares refinement program for macromolecular structures. J. Appl. Cryst.* **22**, 510–516.

Engh, R. A. & Huber, R. (1991). *Accurate bond and angle parameters for X-ray protein structure refinement. Acta Cryst.* A**47**, 392–400.

Fitzgerald, P. M. D., Berman, H., Bourne, P., McMahon, B., Watenpaugh, K. & Westbrook, J. (1996). *The mmCIF dictionary: community review and final approval. Acta Cryst.* A**52** (*Suppl.*), C575.

Fitzgerald, P. M. D., McKeever, B. M., VanMiddlesworth, J. F., Springer, J. P., Heimbach, J. C., Leu, C.-T., Kerber, W. K., Dixon, R. A. F. & Darke, P. L. (1990). *Crystallographic analysis of a complex between human immunodeficiency virus type 1 protease and acetyl-pepstatin at 2.0-A resolution. J. Biol. Chem.* **265**, 14209–14219.

Hall, S. R. (1991). *The STAR file: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The crystallographic information file (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hamilton, W. C. (1965). *Significance tests on the crystallographic R factor. Acta Cryst.* **18**, 502–510.

Hendrickson, W. A. & Konnert, J. H. (1979). *Stereochemically restrained crystallographic least-squares refinement of macromolecule structures.* In *Biomolecular structure, conformation, function and evolution*, edited by R. Srinavisan, Vol. I, pp. 43–57. New York: Pergamon Press.

Hendrickson, W. A. & Lattman, E. E. (1970). *Representation of phase probability distributions for simplified combination of independent phase information. Acta Cryst.* B**26**, 136–143.

Henrick, K., Newman, R., Tagari, M. & Chagoyen, M. (2003). *EMDep: a web-based system for the deposition and validation of high-resolution electron microscopy macromolecular structural information. J. Struct. Biol.* **144**, 228–237.

Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Improved methods for building protein models in electron density maps and the location of errors in these models. Acta Cryst.* A**47**, 110–119.

Leonard, G. A., Hambley, T. W., McAuley-Hecht, K., Brown, T. & Hunter, W. N. (1993). *Anthracycline–DNA interactions at unfavourable base-pair triplet-binding sites: structures of d(CGGCCG)/daunomycin and d(TGGCCA)/adriamycin complexes. Acta Cryst.* D**49**, 458–467.

Luzzati, V. (1952). *Traitement statistique des erreurs dans la determination des structures cristallines. Acta Cryst.* **5**, 802–810.

Narayana, N., Ginell, S. L., Russu, I. M. & Berman, H. M. (1991). *Crystal and molecular structure of a DNA fragment: d(CGTGAATTCACG). Biochemistry*, **30**, 4449–4455.

Shapiro, L., Fannon, A. M., Kwong, P. D., Thompson, A., Lehmann, M. S., Grubel, G., Legrand, J. F., Als-Nielsen, J., Colman, D. R. & Hendrickson, W. A. (1995). *Structural basis of cell–cell adhesion by cadherins. Nature (London)*, **374**, 327–337.

Tickle, I. J., Laskowski, R. A. & Moss, D. S. (1998). *R_{free} and the R_{free} ratio. I. Derivation of expected values of cross-validation residuals used in macromolecular least-squares refinement. Acta Cryst.* D**54**, 547–557.

Ulrich, E. L., Markley, J. L. & Kyogoku, Y. (1989). *Creation of a nuclear magnetic resonance data repository and literature database. Protein Seq. Data Anal.* **2**, 23–37.

Zanotti, G., Berni, R. & Monaco, H. L. (1993). *Crystal structure of liganded and unliganded forms of bovine plasma retinol-binding protein. J. Biol. Chem.* **268**, 10728–10738.

3.7. Classification and use of image data

BY H. J. BERNSTEIN

3.7.1. Introduction

This chapter describes the categories and organization of data items defined in the CBF/imgCIF dictionary. The classification of image data applies to both Crystallographic Binary File (CBF) and Image-supporting Crystallographic Information File (imgCIF) representations. An introduction to CBF data and construction is given in Chapter 2.3. Full details of the CBF/imgCIF dictionary are given in Chapter 4.6.

The main reason for introducing the new items defined in the CBF/imgCIF dictionary was to extend the mmCIF dictionary (Chapter 3.6) to allow the storage of synchrotron diffraction images. However, these items are also important in other fields that use binary image data, including the publication of articles, the creation of web pages and the production of movies.

Data categories in the CBF/imgCIF dictionary can describe one-, two- and three-dimensional array detectors that output data organized by time and/or wavelength. The categories defined at present support modular data that can be extended for future applications without having to make fundamental structural changes. For example, it is anticipated that additional data items will be needed soon to allow higher-dimensional data representations and more complex data structures; these should be accommodated easily.

The CBF/imgCIF dictionary consists of three groups of categories of data items: the ARRAY_DATA group, the AXIS group and the DIFFRN group (Table 3.7.1.1). All fall within the 'Experimental measurements' classification of Table 3.1.10.1. The DIFFRN group already exists in the mmCIF dictionary (Section 3.6.5.2; see also Section 3.2.2.2) and describes the diffraction data and their measurement. Definitions in the CBF/imgCIF dictionary extend and in some cases restate the definitions in the mmCIF dictionary.

The data categories defined in the CBF/imgCIF dictionary are described in this chapter. Table 3.7.1.1 lists the formal category groups declared in the dictionary and the sections of this chapter in which they are discussed. Each section is divided into subsections describing a single category or a small set of closely related categories. Within each subsection, the data names within the relevant categories are listed. Category keys, pointers to parent data items and aliases to data items in the mmCIF dictionary are indicated.

The data collected in an experiment are organized into scans. Each scan consists of one or more frames. Each frame consists of one or more data arrays. The logical data in the data arrays need to be described in terms of physical arrays of image elements. The axes of the laboratory coordinate system needed to describe the physical positions of the image elements and the positioning of the specimen are given in the AXIS category. The axes used for the positioning systems for the specimen and the detector are constructed in the same laboratory coordinate system.

Affiliation: HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA.

Table 3.7.1.1. *Category groups defined in the CBF/imgCIF dictionary*

Section	Category group	Subject covered
Experimental measurements		
3.7.2	ARRAY_DATA	Binary image data
3.7.3	AXIS	Axes required to specify the data collection
3.7.4	DIFFRN	Diffraction experiment

The DIFFRN_DETECTOR_AXIS category relates detector elements to axes. The DIFFRN_MEASUREMENT_AXIS category relates goniometers to axes. The DIFFRN_SCAN_AXIS and DIFFRN_SCAN_FRAME_AXIS categories relate scans to overall axis settings and individual frames to frame-by-frame axis settings, respectively.

The organization of the data in the collected arrays of data is given in the ARRAY_STRUCTURE_LIST category and the physical settings of axes for the centres of pixels that correspond to data points are given in the ARRAY_STRUCTURE_LIST_AXIS category.

3.7.2. Binary image data

The six categories that collectively define the relationship between the sequences of octets in arrays of binary data and the information in the images those octets represent are as follows:

ARRAY_DATA group
The image data (§3.7.2.1)
  ARRAY_DATA
Array elements (§3.7.2.2)
  ARRAY_ELEMENT_SIZE
Intensities (§3.7.2.3)
  ARRAY_INTENSITIES
Organization and encoding of array data (§3.7.2.4)
  ARRAY_STRUCTURE
  ARRAY_STRUCTURE_LIST
  ARRAY_STRUCTURE_LIST_AXIS

3.7.2.1. The image data

Data items in this category are as follows:

ARRAY_DATA
- **_array_data.array_id**
  → **_array_structure.id**
- **_array_data.binary_id**
  _array_data.data

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

Each value of the **_array_data.data** data item is a sequence of octets representing a binary image. **_array_data.array_id** and **_array_data.binary_id**, taken together, uniquely identify each image. The value of **_array_data.array_id** is a pointer to **_array_structure.id** to provide the relationship between the sequence of octets and the logical structure of the image. Since multiple images may have the same logical structure, the purpose of **_array_data.binary_id** is to ensure that each image has a unique identifier.

3.7.2.2. Array elements

Data items in this category are as follows:

ARRAY_ELEMENT_SIZE
- **_array_element_size.array_id**
  → **_array_structure.id**
- **_array_element_size.index**
  → **_array_structure_list.index**

  _array_element_size.size

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

The value of the **_array_element_size.size** data item is a size in metres of an image element (a pixel or voxel). The direction of the measurement is given in each dimension by **_array_element_size.index**. The array structure specifying the organization of the dimensions is referenced by the value of **_array_element_size.array_id**, which is a pointer to **_array_structure.id**. The value of **_array_element_size.index** is a pointer to **_array_structure_list.index**. For data organized into rectangular arrays of pixels or voxels, this gives the spatial dimensions of the individual image elements.

3.7.2.3. Intensities

Data items in this category are as follows:

ARRAY_INTENSITIES
- **_array_intensities.array_id**
  → **_array_structure.id**
- **_array_intensities.binary_id**
  → **_array_data.binary_id**

  _array_intensities.gain
  _array_intensities.gain_esd
  _array_intensities.linearity
  _array_intensities.offset
  _array_intensities.overload
  _array_intensities.scaling
  _array_intensities.undefined_value

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

The relationship between the data values for individual image elements and the number of incident photons can be complex. The data items in the ARRAY_INTENSITIES category provide information about this relationship. The value of **_array_intensities.linearity** states the type of relationship, and the values of **_array_intensities.array_id** and **_array_intensities.binary_id** identify the array structure and the image being discussed. The other items are used in different ways depending on the relationship. If the value of **_array_intensities.linearity** is `raw`, then the image elements hold uninterpreted raw data values from the detector, *e.g.* for calibration. If the value of **_array_intensities.linearity** is `linear`, then the count in an image element is proportional to the incident number of photons by the value of **_array_intensities.gain**. The standard uncertainty (estimated standard deviation) of the gain may be given in **_array_intensities.gain_esd**. The value used for this should be estimated from a good understanding of the physical characteristics of the experimental apparatus. If the value of **_array_intensities.linearity** is `offset`, then the value of **_array_intensities.offset** should be added to the image element value. If the value of **_array_intensities.linearity** is `scaling`, `scaling_offset`, `sqrt_scaled` or `logarithmic_scaled`, the necessary scaling factor is given by the value of **_array_intensities.scaling**. In all cases, the scaling factor is applied to the image element value before the other operations are applied. In the first case, only simple scaling is used. In the second case, the value of **_array_intensities.offset** is added after scaling. In the third case, the scaled value is squared. In the final case, 10 is taken to the power given by the scaled value.

3.7.2.4. Organization and encoding of array data

Data items in these categories are as follows:

(*a*) ARRAY_STRUCTURE
- **_array_structure.id**

  _array_structure.byte_order
  _array_structure.compression_type
  _array_structure.encoding_type

(*b*) ARRAY_STRUCTURE_LIST
- **_array_structure_list.array_id**
- **_array_structure_list.index**
  → **_array_structure.id**

  _array_structure_list.axis_set_id
  _array_structure_list.dimension
  _array_structure_list.direction
  _array_structure_list.precedence

(*c*) ARRAY_STRUCTURE_LIST_AXIS
- **_array_structure_list_axis.axis_id**
  → **_axis.id**
- **_array_structure_list_axis.axis_set_id**
  → **_array_structure_list.axis_set_id**

  _array_structure_list_axis.angle
  _array_structure_list_axis.angle_increment
  _array_structure_list_axis.angular_pitch
  _array_structure_list_axis.displacement
  _array_structure_list_axis.displacement_increment
  _array_structure_list_axis.radial_pitch

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

The data items in the ARRAY_STRUCTURE category show how the stream of octets in a binary image is to be reorganized into words of an appropriate size. Each possible encoding is identified by a value of **_array_structure.id**. In most cases, large images will have been compressed. The type of compression used is given by **_array_structure.compression_type**. Once a stream of octets has been decompressed, it can be organized into words. The type of each word is given by the value of **_array_structure.encoding_type** and the order of mapping octets onto words, most significant octet first ('big-endian') or least significant octet first ('little-endian'), is given by the value of **_array_structure.byte_order**.

The data items in the ARRAY_STRUCTURE_LIST category show how the list of words defined by the ARRAY_STRUCTURE category should be organized into image arrays. The value of **_array_structure_list.array_id** is a pointer to **_array_structure.id**. Each dimension (row, column, sheet *etc.*) of the image is identified by an index, counting from 1, given by **_array_structure_list.index**. The order of nesting of the indices is given by the values of **_array_structure_list.precedence**, with the index of precedence 1 varying most rapidly (*i.e.* having values stored sequentially). The direction of index change for increasing memory location is given by the value of **_array_structure_list.direction**. For a given index, the number of image elements in that direction is given by the value of **_array_structure_list.dimension**.

Data items in the ARRAY_STRUCTURE_LIST_AXIS category describe the physical settings of sets of axes for the centres of pixels that correspond to data points described in the ARRAY_STRUCTURE_LIST category.

In the simplest cases, the physical increments of a single axis correspond to the increments of a single array index. More complex organizations (*e.g.* spiral scans) may require coupled motions along multiple axes.

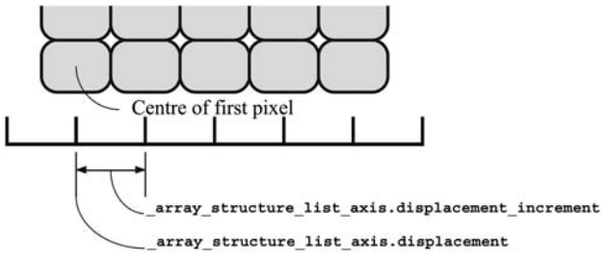


Fig. 3.7.2.1. ARRAY_STRUCTURE_LIST specification of linearly organized image elements.

Note that a spiral scan uses two coupled axes, one for the angular direction and one for the radial direction. This differs from a cylindrical scan for which the two axes are not coupled into one set.

Multiple related axes are gathered together into sets. Each set is identified by the value of the axis set identifier, **_array_structure_list_axis.axis_set_id**, and each axis within a set is identified by the value of **_array_structure_list_axis.axis_id**. Each set given by a value of ***.axis_set_id** is linked to a corresponding value for **_array_structure_list.axis_set_id** to relate settings of the axes in the axis set to particular image elements in ARRAY_STRUCTURE_LIST.

If axes are all independent, no value need be given for **_array_structure_list_axis.axis_set_id**, which is then implicitly given the corresponding value of **_array_structure_list_axis.axis_id**. Each axis given by a value of **_array_structure_list_axis.axis_id** is linked to a corresponding value for **_axis.id** to provide a physical description of the axis. **_array_structure_list_axis.axis_id** and **_array_structure_list_axis.axis_set_id** together uniquely identify a row of data in an ARRAY_STRUCTURE_LIST_AXIS table.

For the remaining data items, there are two important cases to consider: axes that step by Euclidean distance and axes that step by angle. Fig. 3.7.2.1 shows a portion of an array of image elements laid out on a rectangular grid. The starting point of an axis is specified in millimetres by the value of **_array_structure_list_axis.displacement** and the centre-to-centre distance between pixels is specified in millimetres by the value of **_array_structure_list_axis.displacement_increment**.

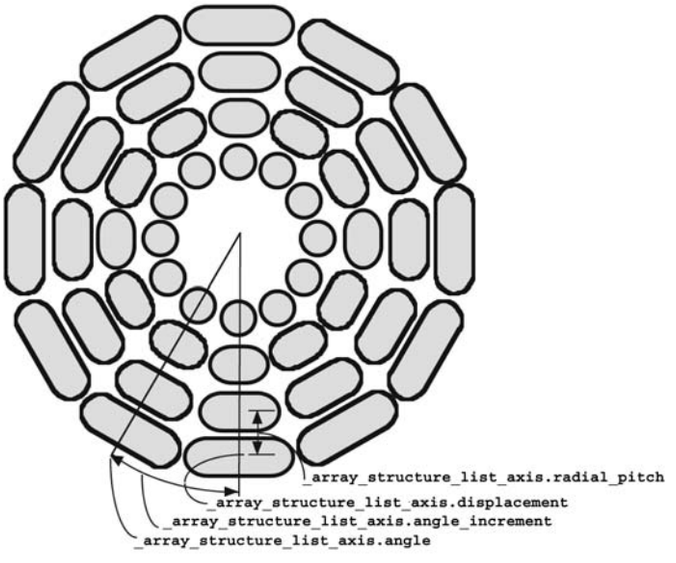


Fig. 3.7.2.2. ARRAY_STRUCTURE_LIST specification of 'constant-angle' image elements in a cylindrical scan. The angular and radial axes are independent. Note that outer-zone image elements are further apart, centre-to-centre, than inner-zone image elements.

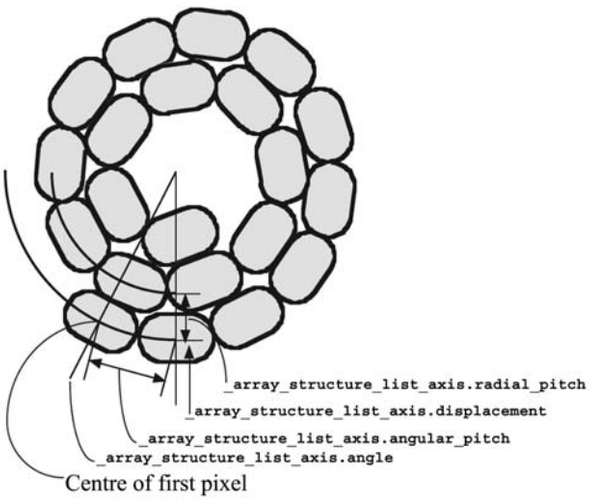


Fig. 3.7.2.3. ARRAY_STRUCTURE_LIST specification of 'constant-velocity' image elements in a cylindrical scan. The angular and radial axes are coupled. Note that outer-zone image elements are the same linear distance apart, centre-to-centre, as the inner-zone image elements.

Fig. 3.7.2.2 shows a portion of an array of image elements laid out in concentric cylinders. The starting point of the angular axis is specified in degrees by the value of **_array_structure_list_axis.angle** and the centre-to-centre angular distance between pixels is specified in degrees by the value of **_array_structure_list_axis.angle_increment**. The starting point of the radial axis is specified by the value of **_array_structure_list_axis.displacement** and the radial distance between cylinders of pixels is specified in millimetres by the value of **_array_structure_list_axis.radial_pitch**. Note that the image elements further from the centre are larger than the image elements closer to the centre.

Fig. 3.7.2.3 shows a portion of a spiral scan array in which the angular and radial axes are coupled. This example represents a 'constant-velocity' scan, in which the size of the image elements does not depend on the distance from the centre. The starting point of the angular axis is again specified in degrees by the value of **_array_structure_list_axis.angle**, but the centre-to-centre distance between pixels is specified in millimetres by the value of **_array_structure_list_axis.angular_pitch**. The coupled radial axis is handled in much the same way as for the uncoupled radial axis in the cylindrical array.

These examples show some of the more common two-dimensional data structures. By coupling an additional axis not in the plane of the first two, regular three-dimensional arrays of data can be represented without additional tags. The categories in the DIFFRN group allow arrays of data to be associated with frames and thereby with time and/or wavelength. More general data structures, for example ones based on dope vectors or hash tables, would require the definition of additional tags, but any data structure (see Aho *et al.*, 1987) that can be handled by a modern computer should be manageable within this framework.

3.7.3. Axes

The category describing the axes required to specify the data collection is as follows:

AXIS group
AXIS

Data items in this category are as follows:

AXIS

- **_axis.equipment**
- **_axis.id**

  _axis.depends_on
  → **_axis.id**

  _axis.offset[1]

  _axis.offset[2]

  _axis.offset[3]

  _axis.type

  _axis.vector[1]

  _axis.vector[2]

  _axis.vector[3]

The bullet (•) *indicates a category key. The arrow* (→) *is a reference to a parent data item.*

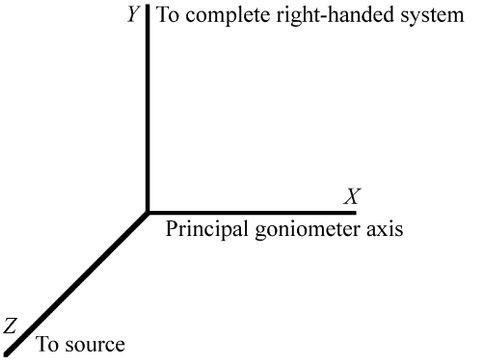


Fig. 3.7.3.1. AXIS laboratory coordinate system. The origin is centred in the specimen.

Data items in the AXIS category record the information required to describe the goniometer, detector, source and other axes needed to specify a data collection. The location of each axis is specified by two vectors: the axis itself, given as a unit vector, and an offset to the base of the unit vector. These vectors are referenced to a right-handed laboratory coordinate system with its origin in the specimen, as shown in Fig. 3.7.3.1.

The *X* axis is aligned to the mechanical axis pointing from the specimen along the principal axis of the goniometer.

The *Z* axis is defined next. The *Z* axis is derived from the source axis (the axis running from the sample to the source). If the source axis is orthogonal to the *X* axis, the source axis is the *Z* axis. If the source axis is not orthogonal to the *X* axis, the *Z* axis is the component of the source axis orthogonal to the *X* axis. The direction is chosen to form an acute angle with the source axis.

The *Y* axis is defined last. The *Y* axis completes an orthogonal right-handed system defined by the *X* axis and the *Z* axis (see below).

These axes are based on the goniometer, not on the orientation of the detector, gravity *etc.* The vectors necessary to specify all other axes are given by sets of three components in the order (*X*, *Y*, *Z*). If the axis involved is a rotation axis, it is right-handed, *i.e.* as one views the object to be rotated from the origin (the tail) of the unit vector, the rotation is clockwise. If a translation axis is specified, the direction of the unit vector specifies the sense of positive translation.

Note: This choice of coordinate system is similar to but significantly different from the choice in *MOSFLM* (Lesley & Powell, 2003), in which *X* is along the X-ray beam (our *Z* axis) and *Z* is along the rotation axis.

All rotations are given in degrees and all translations are given in millimetres.

Axes may be dependent on one another. The *X* axis is the only axis that is strictly connected to the hardware. All other axes are specified by the positions they would assume when the axes upon which they depend are at their zero points.

When specifying detector axes, each axis is specified relative to the beam centre. The location of the beam centre on the detector should be given in the DIFFRN_DETECTOR category in millimetres from the (0, 0) corner of the detector and should be corrected for distortion.

It should be noted that many different origins arise in the definition of an experiment. In particular, as noted above, we need to specify the location of the beam centre on the detector in terms of the origin of the detector, which is, of course, not coincident with the centre of the sample.

Each axis is uniquely identified by the values of **_axis.id** and of **_axis.equipment**. An axis may be a translation axis, a rotation axis or an axis for which the mode of motion is not relevant. The type of axis is specified by the value of **_axis.type**. The base of the axis is specified by the point in the laboratory coordinate system given by the values of **_axis.offset[1]**, **_axis.offset[2]** and **_axis.offset[3]**, and the direction of the axis from that base, as a dimensionless unit vector, is given by **_axis.vector[1]**, **_axis.vector[2]** and **_axis.vector[3]**.

3.7.4. The diffraction experiment

The categories relating to the diffraction experiment are as follows:

DIFFRN group

Frames of data (§3.7.4.1)

DIFFRN_DATA_FRAME

The detector apparatus (§3.7.4.2)

DIFFRN_DETECTOR

DIFFRN_DETECTOR_AXIS

DIFFRN_DETECTOR_ELEMENT

Apparatus and instrumentation at the crystal (§3.7.4.3)

DIFFRN_MEASUREMENT

DIFFRN_MEASUREMENT_AXIS

The radiation source (§3.7.4.4)

DIFFRN_RADIATION

Intensity measurements (§3.7.4.5)

DIFFRN_REFLN

Diffraction scans (§3.7.4.6)

DIFFRN_SCAN

DIFFRN_SCAN_AXIS

DIFFRN_SCAN_FRAME

DIFFRN_SCAN_FRAME_AXIS

The CBF/imgCIF dictionary extends the mmCIF categories in the DIFFRN group, which are very similar to their corresponding categories in the core CIF dictionary. The DIFFRN group is introduced in the description of the core CIF dictionary in Section 3.2.2.2. Its use in the mmCIF dictionary is described in Section 3.6.5.2, from which we quote: 'The categories in the DIFFRN category group describe the diffraction experiment. Data items in the DIFFRN category itself can be used to give overall information about the experiment, such as the temperature and pressure. Examples of the other categories are DIFFRN_DETECTOR, which is used for describing the detector used for data collection, and DIFFRN_SOURCE, which is used to give details of the source of the radiation used in the experiment. Data items in the DIFFRN_REFLN category can be used to give information about the raw data and data items in the DIFFRN_REFLNS category can be used to give information about all the reflection data collectively.' In this chapter we focus on the CBF/imgCIF extensions.

3.7.4.1. Frames of data

Data items in this category are as follows:

DIFFRN_DATA_FRAME
- **_diffrn_data_frame.detector_element_id**
  → **_diffrn_detector_element.id**
- **_diffrn_data_frame.id**
  _diffrn_data_frame.array_id
  → **_array_structure.id**
  _diffrn_data_frame.binary_id
  → **_array_data.binary_id**

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

Data items in the DIFFRN_DATA_FRAME category record details about each frame of data. An experiment may produce multiple frames of data and each frame may be constructed from data provided by multiple detector elements. Each complete frame of data is uniquely identified by the value of **_diffrn_data_frame.id**. The detector elements used are specified by values of **_diffrn_data_frame.detector_element_id**, which forms the category key together with **_diffrn_data_frame.id**. **_diffrn_data_frame.detector_element_id** is a pointer to **_diffrn_detector_element.id** in the DIFFRN_DETECTOR_ELEMENT category. The structure of the data in the frame is completed by giving values for **_diffrn_data_frame.array_id** (a pointer to **_array_structure.id**). The particular blocks of data in the frame are specified by giving values of **_diffrn_data_frame.binary_id** (a pointer to **_array_data.binary_id**).

3.7.4.2. The detector apparatus

Data items in these categories are as follows:

(*a*) DIFFRN_DETECTOR
- **_diffrn_detector.diffrn_id**
  → **_diffrn.id**
- **_diffrn_detector.id**
  _diffrn_detector.details
  _diffrn_detector.detector
  _diffrn_detector.dtime
  _diffrn_detector.number_of_axes
  _diffrn_detector.type

(*b*) DIFFRN_DETECTOR_AXIS
- **_diffrn_detector_axis.axis_id**
  → **_axis.id**
- **_diffrn_detector_axis.detector_id**
  → **_diffrn_detector.id**

(*c*) DIFFRN_DETECTOR_ELEMENT
- **_diffrn_detector_element.id**
- **_diffrn_detector_element.detector_id**
  → **_diffrn_detector.id**
  _diffrn_detector_element.center[1]
  _diffrn_detector_element.center[2]

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics are defined in the mmCIF dictionary.

The DIFFRN_DETECTOR category is defined in the mmCIF dictionary (Section 3.6.5.2; see the detailed discussion in Section 3.2.2.2.4). The CBF/imgCIF dictionary restates the DIFFRN_DETECTOR category, adding new tags. Data items in the DIFFRN_DETECTOR category describe the detector used to measure the scattered radiation, including any analyser and post-sample collimation. In order to allow for multiple detectors, the category key has been extended to include **_diffrn_detector.id** to uniquely identify each detector. If there is only one detector, **_diffrn_detector.id** need not be specified, and it will implicitly default to the value of **_diffrn_detector.diffrn_id** (a pointer to **_diffrn.id** in the DIFFRN category in the mmCIF dictionary). The general class of detector is given by the value of **_diffrn_detector.detector** with the make and model given by the value of **_diffrn_detector.type**. Any special aspects of the detector not covered elsewhere are given by the value of **_diffrn_detector.details**. As in mmCIF, the value of **_diffrn_detector.dtime** gives the deadtime of the detector. Additional data items may need to be added in the future for complex inhomogeneous deadtime situations. In addition, the number of axes can be specified using **_diffrn_detector.number_of_axes**.

Data items in the DIFFRN_DETECTOR_AXIS category associate axes with detectors. Each axis is associated with a detector through the value of **_diffrn_detector_axis.detector_id** (a pointer to **_diffrn_detector.id**). The value of ***.axis_id** (a pointer to **_axis.id**) identifies an axis. Together ***.detector_id** and ***.axis_id** form the category key.

Data items in the DIFFRN_DETECTOR_ELEMENT category record details about the spatial layout and other characteristics of each element of a detector which may have multiple elements, giving the *X* and *Y* coordinates of the position of the beam centre relative to the lower left corner of each detector element. Each detector element is identified by the value of **_diffrn_detector_element.id** and the detector of which it is an element is identified by the value of **_diffrn_detector_element.detector_id** (a pointer to **_diffrn_detector.id**).

In most cases, it would be preferable to use the more detailed information provided in the ARRAY_STRUCTURE_LIST and ARRAY_STRUCTURE_LIST_AXIS categories rather than simply specifying the coordinates of the centre of the beam relative to the lower left corner of each detector element.

3.7.4.3. Apparatus and instrumentation at the crystal

Data items in these categories are as follows:

(*a*) DIFFRN_MEASUREMENT
- ***_diffrn_measurement.diffrn_id***
  → **_diffrn.id**
- ***_diffrn_measurement.device***
- **_diffrn_measurement.id**
  _diffrn_measurement.details
  _diffrn_measurement.device_details
  _diffrn_measurement.device_type
  _diffrn_measurement.method
  _diffrn_measurement.number_of_axes
  _diffrn_measurement.specimen_support

(*b*) DIFFRN_MEASUREMENT_AXIS
- **_diffrn_measurement_axis.axis_id**
  → **_axis.id**
- **_diffrn_measurement_axis.measurement_device**
  → **_diffrn_measurement.device**
- **_diffrn_measurement_axis.measurement_id**
  → **_diffrn_measurement.id**

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics are defined in the mmCIF dictionary.

The DIFFRN_MEASUREMENT category is defined in the mmCIF dictionary (Section 3.6.5.2; see the detailed discussion in Section 3.2.2.2.3). The CBF/imgCIF dictionary restates the DIFFRN_MEASUREMENT category, adding new tags. Data items in the DIFFRN_MEASUREMENT category record details about the device used to orient and/or position the crystal during data measurement and the manner in which the diffraction data were measured. To allow for multiple measurement devices, **_diffrn_measurement.id** has been added to the category key. The number of axes is given by the value of **_diffrn_measurement.number_of_axes**. The axes should be described using entries in DIFFRN_MEASUREMENT_AXIS.

Data items in the DIFFRN_MEASUREMENT_AXIS category associate axes with goniometers, just as data items in the DIFFRN_DETECTOR_AXIS category associate axes with detectors.

3.7.4.4. The radiation source

Data items in this category are as follows:

DIFFRN_RADIATION
- **_diffrn_radiation.diffrn_id**
  → **_diffrn.id**

_diffrn_radiation.collimation
_diffrn_radiation.div_x_source
_diffrn_radiation.div_y_source
_diffrn_radiation.div_x_y_source
_diffrn_radiation.filter_edge
_diffrn_radiation.inhomogeneity
_diffrn_radiation.monochromator
_diffrn_radiation.polarisn_norm
_diffrn_radiation.polarisn_ratio
_diffrn_radiation.polarizn_source_norm
_diffrn_radiation.polarizn_source_ratio
_diffrn_radiation.probe
_diffrn_radiation.type
_diffrn_radiation.wavelength_id
→ **_diffrn_radiation_wavelength.id**
_diffrn_radiation.xray_symbol

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics are defined in the mmCIF dictionary.

The DIFFRN_RADIATION category is defined in the mmCIF dictionary (Section 3.6.5.2; see the detailed discussion in Section 3.2.2.2.2). The CBF/imgCIF dictionary adds the items **_diffrn_radiation.div_x_source**, ***.div_y_source** and ***.div_x_y_source** to specify beam crossfire, and the items **_diffrn_radiation.polarizn_source_norm** and ***.polarizn_source_ratio** to provide a definition of polarization relative to the laboratory coordinate system rather than relative to the diffraction plane. The value of the beam crossfire component **_diffrn_radiation.div_x_source** is the mean deviation in degrees of the X-ray beam from being parallel to the X axis as it illuminates the sample. The value of the beam crossfire component **_diffrn_radiation.div_y_source** is the mean deviation in degrees of the X-ray beam from being parallel to the Y axis as it illuminates the sample. The value of the beam crossfire component **_diffrn_radiation.div_x_y_source** is the correlation of the X and Y components. The value of the normal component of the polarization **_diffrn_radiation.polarizn_source_norm** is the angle in degrees, as viewed from the specimen, between the normal to the polarization plane and the laboratory Y axis as defined in the AXIS category. The dimensionless value of **_diffrn_radiation.polarisn_ratio** is the ratio $(I_p - I_n)/(I_p + I_n)$, where I_n is the intensity (amplitude squared) of the electric vector of the illumination of the sample normal to the polarization and I_p is the intensity of the electric vector of the illumination of the sample in the plane of polarization. With suitable choices of laboratory axes, the definitions conform to synchrotron conventions. See Chapter 4.6 for a detailed description of these items.

3.7.4.5. Intensity measurements

Data items in this category are as follows:

DIFFRN_REFLN
- **_diffrn_refln.frame_id**
  → **_diffrn_data_frame.id**
- **_diffrn_refln.id**
- **_diffrn_refln.diffrn_id**

_diffrn_refln.angle_chi
_diffrn_refln.angle_kappa
_diffrn_refln.angle_omega
_diffrn_refln.angle_phi
_diffrn_refln.angle_psi
_diffrn_refln.angle_theta
_diffrn_refln.attenuator_code
_diffrn_refln.counts_bg_1
_diffrn_refln.counts_bg_2
_diffrn_refln.counts_net
_diffrn_refln.counts_peak
_diffrn_refln.counts_total
_diffrn_refln.detect_slit_horiz
_diffrn_refln.detect_slit_vert
_diffrn_refln.elapsed_time
_diffrn_refln.index_h
_diffrn_refln.index_k
_diffrn_refln.index_l
_diffrn_refln.intensity_net
_diffrn_refln.intensity_sigma
_diffrn_refln.scale_group_code
_diffrn_refln.scan_mode
_diffrn_refln.scan_mode_backgd
_diffrn_refln.scan_rate
_diffrn_refln.scan_time_backgd
_diffrn_refln.scan_width
_diffrn_refln.sint_over_lambda
_diffrn_refln.standard_code
_diffrn_refln.wavelength
_diffrn_refln.wavelength_id

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics are defined in the mmCIF dictionary.

The DIFFRN_REFLN category is defined in the mmCIF dictionary (Section 3.6.5.2; see the detailed discussion in Section 3.2.2.2.2). Data items in the DIFFRN_REFLN category record details of the intensities measured in the diffraction data set identified by **_diffrn_refln.diffrn_id**. The CBF/imgCIF dictionary extends the key with **_diffrn_refln.frame_id** (a pointer to **_diffrn_data_frame.id**), so that multiple data sets may be recorded.

3.7.4.6. Diffraction scans

Data items in these categories are as follows:

(*a*) DIFFRN_SCAN
- **_diffrn_scan.id**

_diffrn_scan.date_end
_diffrn_scan.date_start
_diffrn_scan.frame_id_start
→ **_diffrn_data_frame.id**
_diffrn_scan.frame_id_end
→ **_diffrn_data_frame.id**
_diffrn_scan.frames
_diffrn_scan.integration_time

(*b*) DIFFRN_SCAN_AXIS
- **_diffrn_scan_axis.axis_id**
  → **_axis.id**
- **_diffrn_scan_axis.scan_id**
  → **_diffrn_scan.id**

_diffrn_scan_axis.angle_start
_diffrn_scan_axis.angle_range
_diffrn_scan_axis.angle_increment
_diffrn_scan_axis.angle_rstrt_incr
_diffrn_scan_axis.displacement_start
_diffrn_scan_axis.displacement_range
_diffrn_scan_axis.displacement_increment
_diffrn_scan_axis.displacement_rstrt_incr

(*c*) DIFFRN_SCAN_FRAME
_diffrn_scan_frame.date
- **_diffrn_scan_frame.frame_id**
  → **_diffrn_data_frame.id**
- **_diffrn_scan_frame.scan_id**
  → **_diffrn_scan.id**

_diffrn_scan_frame.frame_number
_diffrn_scan_frame.integration_time

(*d*) DIFFRN_SCAN_FRAME_AXIS
- **_diffrn_scan_frame_axis.axis_id**
  → **_axis.id**
- **_diffrn_scan_frame_axis.frame_id**
  → **_diffrn_data_frame.id**

_diffrn_scan_frame_axis.angle

```
_diffrn_scan_frame_axis.angle_increment
_diffrn_scan_frame_axis.angle_rstrt_incr
_diffrn_scan_frame_axis.displacement
_diffrn_scan_frame_axis.displacement_increment
_diffrn_scan_frame_axis.displacement_rstrt_incr
```

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item.

Data items in the DIFFRN_SCAN category describe the parameters of one or more scans, relating axis positions to frames. Each scan is uniquely identified by the value of **_diffrn_scan.id**. The data items in this category give overall information for the scan. The detailed frame-by-frame data are given in DIFFRN_SCAN_FRAME and DIFFRN_SCAN_FRAME_AXIS. The values of **_diffrn_scan.date_start** and ***.date_end** give the starting and ending time for a scan. The original definition of the *yyyy-mm-dd* data type, which includes date and time, has been extended in the CBF/imgCIF dictionary. This allows the seconds part of the time to include an optional decimal fraction. The approximate average integration time for each step of the scan is given by the value of **_diffrn_scan.integration_time**. The scan is tied to individual frame IDs by the values of **_diffrn_scan.frame_id_start** and ***.frame_id_end**. The number of frames in the scan is given by the value of **_diffrn_scan.frames**.

Data items in the DIFFRN_SCAN_AXIS category describe the settings of axes for particular scans. Unspecified axes are assumed to be at their zero points. The vector of each axis is not given here, because it is provided in the AXIS category. By making **_diffrn_scan_axis.scan_id** and **_diffrn_scan_axis.axis_id** keys of the DIFFRN_SCAN_AXIS category, an arbitrary number of scanning and fixed axes can be specified for a scan. The value of **_diffrn_scan_axis.scan_id** (a pointer to **_diffrn_scan.id**) identifies the scan and the values of **_diffrn_scan_axis.axis_id** (a pointer to **_axis.id**) associate particular axes with that scan. The steps of each axis are specified by ***_start**, ***_range**, ***_increment** and ***_rstrt_incr** values for angles or for displacements. The ***_start** value is the setting of the relevant axis at the start of the scan. The ***_range** value is the total change in the axis setting through the scan. The ***_increment** value is the increment in the axis setting for each step of the scan. The ***_rstrt_incr** value is the increment in the axis setting after each step of the scan.

Data items in the DIFFRN_SCAN_FRAME category describe the relationship of particular frames to scans. The value of **_diffrn_scan_frame.frame_id** (a pointer to **_diffrn_data_frame.id**) identifies the frame. The value of **_diffrn_scan_frame.scan_id** (a pointer to **_diffrn_scan.id**) identifies the scan of which the frame is a part. Together **_diffrn_scan_frame.frame_id** and ***.scan_id** form the category key. The value of **_diffrn_scan_frame.date** gives the date and time of the start of the data collection for the frame. The value of **_diffrn_scan_frame.frame_number** gives the number of the frame (starting with 1). The value of **_diffrn_scan_frame.integration_time** gives the precise time in seconds to integrate this step of the scan.

Table A3.7.1.1. *Categories in the CBF/imgCIF dictionary*
Numbers in parentheses refer to the section of this chapter in which each category is described in detail.

ARRAY_DATA group (§3.7.2)	DIFFRN_DETECTOR_AXIS (§3.7.4.2(*b*))
ARRAY_DATA (§3.7.2.1)	DIFFRN_DETECTOR_ELEMENT (§3.7.4.2(*c*))
ARRAY_ELEMENT_SIZE (§3.7.2.2)	DIFFRN_MEASUREMENT (§3.7.4.3(*a*))
ARRAY_INTENSITIES (§3.7.2.3)	DIFFRN_MEASUREMENT_AXIS (§3.7.4.3(*b*))
ARRAY_STRUCTURE (§3.7.2.4(*a*))	DIFFRN_RADIATION (§3.7.4.4)
ARRAY_STRUCTURE_LIST (§3.7.2.4(*b*))	DIFFRN_REFLN (§3.7.4.5)
ARRAY_STRUCTURE_LIST_AXIS (§3.7.2.4(*c*))	DIFFRN_SCAN (§3.7.4.6(*a*))
AXIS group (§3.7.3)	DIFFRN_SCAN_AXIS (§3.7.4.6(*b*))
AXIS (§3.7.3)	DIFFRN_SCAN_FRAME (§3.7.4.6(*c*))
DIFFRN group (§3.7.4)	DIFFRN_SCAN_FRAME_AXIS (§3.7.4.6(*d*))
DIFFRN_DATA_FRAME (§3.7.4.1)	
DIFFRN_DETECTOR (§3.7.4.2(*a*))	

Data items in the DIFFRN_SCAN_FRAME_AXIS category describe the settings of axes for particular frames. Unspecified axes are assumed to be at their zero points. If for any given frame non-zero values apply for any of the data items in this category, those values should be given explicitly in this category and not simply inferred from values in DIFFRN_SCAN_AXIS. Since the collection for a given frame may involve multiple axes, the frame involved is identified by the value of **_diffrn_scan_frame_axis.frame_id** (a pointer to **_diffrn_data_frame.id**) and each axis is identified by the value of **_diffrn_scan_frame_axis.axis_id** (a pointer to **_axis.id**). Together **_diffrn_scan_frame_axis.frame_id** and ***.axis_id** form the category key. If the axis is an axis of rotation, the axis settings for the frame are given by the values of **_diffrn_scan_frame_axis.angle**, ***.angle_increment** and ***.angle_rstrt_incr**. If the axis is a translation axis, the axis settings for the frame are given by the values of **_diffrn_scan_frame_axis.displacement**, ***.displacement_increment** and ***.displacement_rstrt_incr**. The integration begins at the setting given by the value of **_diffrn_scan_frame_axis.angle** or of ***.displacement**. The ***_increment** value gives the change of axis setting during the scan. At the end of the integration, the axis may need to be repositioned by an additional amount. That amount is given by ***_rstrt_incr**.

Appendix 3.7.1
Category structure of the CBF/imgCIF dictionary

Table A3.7.1.1 provides an overview of the structure of the CBF/imgCIF dictionary by category group and member categories.

We are grateful to Frances C. Bernstein, Paula Fitzgerald and Bob Sweet for their helpful comments and suggestions.

References

Aho, A. V., Hopcroft, J. E. & Ullman, J. D. (1987). *Data structures and algorithms.* Reading, MA: Addison–Wesley.

Leslie, A. G. W. & Powell, H. (2003). *MOSFLM V6.11*. http://www.ccp4.ac.uk/dist/x-windows/Mosflm.

3.8. Classification and use of symmetry data

BY I. D. BROWN

3.8.1. Introduction

Symmetry lies at the heart of crystallography. Volume A of *International Tables for Crystallography* (2002) (*IT* A) is among the most widely consulted crystallographic texts, since it defines the concepts underlying the theory of three-dimensional crystal symmetry and gives detailed descriptions of the 230 different space groups. These descriptions are essential for anyone studying crystals, since no structure report is complete if the space group is not given.

For this reason, the original core CIF dictionary included a small number of the essential items needed to describe the symmetry of a crystal structure. Among these were three different forms of the space-group name (symbol) and a listing of the symmetry operations in algebraic form. This group of items has proved sufficient for the purpose of archiving crystal structures, but it does not readily lend itself to further extension.

Symmetry concepts have precise definitions, which makes them particularly amenable to manipulation by computer. It is therefore natural to consider the ways in which the concepts of space-group theory and practice should be represented in CIF. Possible applications for an extended list of symmetry items in CIF include online access to the tables given in *International Tables for Crystallography* Volume A (properties of space groups), Volume A1 (relationships between space groups) and Volume E (properties of layer and frieze groups), as well as to allow the description of higher-dimensional symmetry in CIFs reporting quasicrystal, magnetic and modulated structures. To be effective, any CIF definitions have to reflect the current state of space-group theory, as well as meeting the needs of future software that will automatically assemble and manipulate information drawn from crystallographic and other databases.

3.8.2. Dictionary design considerations

In 1995, a working group was established to draw up a CIF symmetry dictionary. The group quickly encountered two problems. Firstly, scientists in different fields have different expectations of space-group theory and conflicting requirements for CIF definitions, and secondly the familiar presentation of space groups that works well in a printed book such as *IT* A is not necessarily the best way of presenting the material in a computer-based system. These problems are illustrated by the difficulties that the working group encountered in defining the space-group symbols. Each space group can be fully described only within a particular coordinate system or setting, but most space groups can be represented in many different settings and no one setting is uniquely fitted for this role. Ideally, one would like to have access to representations of the symmetry operations in all possible settings, but this is impractical in a printed book. *IT* A restricts itself to representing the space groups in a small number, often only one, of the many possible settings. However, computers are not limited in the same way, since it is possible to calculate the operations for any desired setting without additional overhead. This is an advantage for experimental crystallographers who like to choose the setting to suit the crystal, but theorists prefer to use the same fixed reference setting for all instances of a space group because it leads to a consistent description of group–subgroup and other relationships between the space groups.

This difficulty is resolved in the CIF symmetry dictionary by recognizing that there are two types of space-group symbol: those such as the Schoenflies symbol and the *International Tables* number that identify the space group without specifying the setting; and those such as the Hall symbol (Hall & Grosse-Kunstleve, 2001) that are symmetry generators and therefore define the space group within a particular setting. The weakness of the setting-independent symbols is that they are labels that carry at best only a limited amount of information about the symmetry. Full details of the symmetry operations can only be found by consulting a concordance that lists the operations in some arbitrarily chosen setting. The weakness of the symmetry generators is that different settings of the same space group require different symbols, making it difficult to recognize when two symbols refer to the same space group. Unfortunately, the widely used Hermann–Mauguin symbols suffer from both weaknesses. They can be used as space-group generators, but they do not cover all possible settings because they contain no information about the choice of origin. This makes them inappropriate for use when all possible settings are needed. But equally they cannot be used as unique space-group identifiers, since many space groups can be described by more than one Hermann–Mauguin symbol. They are not therefore well adapted to computer use, but they are popular because they are easy to interpret and they indicate in a simple way the relationship between the symmetry operations and the axis system.

Because CIF requires a unique and precise definition for any symbol that a program is required to interpret, the working group found it necessary to define different versions of the Hermann–Mauguin symbol. There is a tightly defined version, `_space_group.name_H-M_ref`, which can be used to generate the symmetry operations of the reference setting, but only one form of this symbol (that given in the list of values specified by the dictionary) can be used for a given space group. For consistency with earlier work, the reference settings in this list have all been chosen from among those listed in *IT* A. Where there is a choice of settings in *IT* A, the *b*-axis setting is chosen for monoclinic, and the hexagonal setting is chosen for rhombohedral space groups. Where two different origin choices are given, the second (origin at a centre of symmetry) is chosen. A second more loosely defined Hermann–Mauguin symbol, `_space_group.name_H-M_alt`, is defined for the benefit of those who like to use the Hermann–Mauguin symbol to indicate the axis system adopted, but as this item does not have a precise definition, it should not be interpreted by a software application. It should be treated as a text string that can be displayed for the benefit of the user.

The symmetry CIF dictionary differs in an important way from the other CIF dictionaries. While the other dictionaries define items that give the results of an experimental measurement, all the items in the symmetry dictionary are either theoretically derivable or, like the reference settings and Wyckoff letters, are arbitrarily

Affiliation: I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1.

assigned. The addition of algorithmic methods in future versions of CIF dictionaries will allow many of the symmetry items to be calculated as needed from expressions included in the dictionary as long as a space-group generator, such as the Hall symbol, is present in the CIF.

The symmetry CIF dictionary is written in DDL2 (Chapter 2.6) as this allows the relationships between the items to be given explicitly. Version 1.0 of the dictionary contains CIF definitions for all the basic symmetry items needed to describe a three-dimensional space group as described in more detail in Section 3.8.3. Some of these items have been incorporated in the core CIF dictionary, in which they have been converted to DDL1 (Chapter 2.5) to match the dictionary definition language of the core CIF dictionary. While the items in the symmetry CIF dictionary are designed in part to replace those in the symmetry category of the original core CIF dictionary, they are defined in a way that allows a CIF to include descriptions of several space groups in several settings. A single CIF could, in principle, include the symmetry operations and Wyckoff positions of all possible settings of the 230 space groups.

3.8.3. Arrangement of the dictionary

The three categories in version 1.0 of the symmetry CIF dictionary all lie within the SPACE_GROUP category group and are classified among the category groups defining the structural model listed in Section 3.1.10. (For convenience, in Chapters 3.2 and 3.4 the **_space_group_*** items introduced from the symmetry dictionary to the core and modulated structures dictionaries are discussed within an informal DDL1 SYMMETRY category group.)

The categories in the symmetry CIF dictionary are:

SPACE_GROUP
SPACE_GROUP_SYMOP
SPACE_GROUP_WYCKOFF

The first describes the properties of the space group as a whole, the second describes the properties of the symmetry operations and the third describes the properties of the special positions. The three categories are linked by a space-group identifier which allows items looped in the last two categories to be related back to one of the space groups defined in the first.

Data items in these categories are as follows:

(*a*) SPACE_GROUP

- ***_space_group.id***
  _space_group.Bravais_type
  _space_group.centring_type
  _space_group.crystal_system
  _space_group.Laue_class

  _space_group.IT_number
  _space_group.name_Hall
  _space_group.name_H-M_alt
  _space_group.name_H-M_alt_description
  _space_group.name_H-M_full
  _space_group.name_H-M_ref
  _space_group.name_Schoenflies
  _space_group.Patterson_name_H-M
  _space_group.point_group_H-M
  _space_group.reference_setting
  _space_group.transform_Pp_abc
  _space_group.transform_Qq_xyz

(*b*) SPACE_GROUP_SYMOP

- ***_space_group_symop.id***
  _space_group_symop.generator_xyz
  _space_group_symop.operation_description
  _space_group_symop.operation_xyz
  _space_group_symop.sg_id
  → **_space_group.id**

(*c*) SPACE_GROUP_WYCKOFF

- **_space_group_Wyckoff.id**
  _space_group_Wyckoff.coords_xyz
  _space_group_Wyckoff.letter
  _space_group_Wyckoff.multiplicity
  _space_group_Wyckoff.sg_id
  → **_space_group.id**
  _space_group_Wyckoff.site_symmetry

The bullet (•) indicates a category key. The arrow (→) is a reference to a parent data item. Items in italics are also found in the core CIF dictionary.

The information contained in the SPACE_GROUP category relates to the properties of the space group as a whole. Three different kinds of properties are defined here.

Firstly, the names (or symbols) used to describe the space group are defined. As mentioned above, these divide themselves into ones that identify the space group without specifying any particular setting, and ones that can be used to generate the symmetry operations and therefore also specify the setting. Because of the ambiguities involved in the Hermann–Mauguin symbol, three different versions are defined with different degrees of rigour. **_space_group.name_H-M_ref** may only include the Hermann–Mauguin symbol of the reference setting. **_space_group.name_H-M_alt** and **_space_group.name_H-M_full** give the user the freedom to give the symbol in any setting, but cannot be reliably interpreted by a computer.

Secondly, the SPACE_GROUP category contains information about the symmetry properties of the space group, such as its Laue class, Bravais type and point group.

Thirdly, the SPACE_GROUP category contains information which specifies the setting. Although this is implicit in the Hall symbol or in the list of symmetry operations that are given in the SPACE_GROUP_SYMOP category, it can be made explicit by including the transformation needed to generate the setting used in the CIF from the reference setting specified in the dictionary. The reference setting is defined in two ways: firstly, in the list of allowed values of **_space_group.name_H-M_ref**; and secondly in a concordance correlating the *International Tables* number and the Schoenflies symbol of the space group with the Hermann–Mauguin symbol of the reference setting and the Hall symbol. Either of the latter two can be used to generate the symmetry operations in the reference setting.

Information on several space groups may be looped. In this case, each space group is identified by the item **_space_group.id**, which is a parent to various ***.sg_id** items in the other categories. This allows a number of different space groups, or different settings of the same space group, to be defined within the same CIF.

Although the most elegant way of specifying the symmetry operations of the space group is to use the Hermann–Mauguin symbol of the reference setting or the Hall symbol (depending on the setting), it is common practice to list all the symmetry operations explicitly in a CIF. For each space group these must appear in a loop and so require their own category, SPACE_GROUP_SYMOP. The symmetry operations may be specified in one of two ways, either through a full list of all the operations of the group or through a restricted list of generators which, when multiplied by each other, generate the full list.

The list of symmetry operations may contain the operations of several space groups, the particular space group being identified by **_space_group_symop.sg_id**.

Special positions are looped in the SPACE_GROUP_WYCKOFF category, which permits a description of the properties of each special position of one or more space groups. In the current structure of CIF it is not possible to give all the equivalent positions associated with a particular special position, but these can easily

be generated by applying the symmetry operations of the space group to the representative special position whose coordinates are included. Although the multiplicity and site symmetry of a given special position can be calculated if the symmetry operations are known, the Wyckoff letter cannot be calculated since it is assigned arbitrarily and is setting-independent.

As with the symmetry operations, it is possible to include the special positions of more than one space group, each space group being identified by `_space_group_Wyckoff.sg_id`.

3.8.4. Future developments

Version 1.0 of the symmetry CIF dictionary contains only the basic items needed to define the properties of the three-dimensional space groups. It is only the first step in the definition of a more comprehensive symmetry dictionary. Future versions will define items needed to describe the higher-dimensional symmetries that have recently proved popular for describing modulated structures and quasicrystals, and items to make it easier for computers to explore the relationships between a space group and its super- and subgroups.

The addition of methods in future versions of the DDL will permit the dictionary to include the expressions for calculating each item from other items in the CIF. Methods will be particularly useful in the symmetry CIF dictionary, since almost all its items can be calculated from a single item: a symmetry generator name. The symmetry CIF dictionary is an essential component in the plan to make the suite of CIF dictionaries a self-contained online compendium of crystallographic knowledge.

It is my pleasure to acknowledge the contributions made by members of the working group that prepared the dictionary: Ralf Grosse-Kunstleve, Vojtech Kopsky, Danny Litvin, Gotzon Madariaga, Mois Aroyo, Uri Shmueli, Ad Thiers and Hans Wondratschek.

References

Hall, S. R. & Grosse-Kunstleve, R. W. (2001). *International tables for crystallography*, Vol. B, *Reciprocal space*, edited by U. Shmueli, 2nd ed., Appendix A1.4.2.3. Dordrecht: Kluwer Academic Publishers.

International Tables for Crystallography (2002). Vol. A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

4. DATA DICTIONARIES

BY F. H. ALLEN, J. M. BARNARD, H. M. BERMAN, H. J. BERNSTEIN, P. E. BOURNE, I. D. BROWN, A. P. F. COOK, P. M. D. FITZGERALD, S. R. HALL, A. P. HAMMERSLEY, B. MCMAHON, G. MADARIAGA, P. R. MALLINSON, B. H. TOBY, K. D. WATENPAUGH AND J. D. WESTBROOK

4.1. Core dictionary (coreCIF) (S. R. HALL, F. H. ALLEN AND I. D. BROWN) 210
4.2. Powder dictionary (pdCIF) (B. H. TOBY) 258
4.3. Modulated and composite structures dictionary (msCIF) (G. MADARIAGA) 270
4.4. Electron density dictionary (rhoCIF) (P. R. MALLINSON) 290
4.5. Macromolecular dictionary (mmCIF) (P. M. D. FITZGERALD, J. D. WESTBROOK, P. E. BOURNE, B. MCMAHON, K. D. WATENPAUGH AND H. M. BERMAN) 295
4.6. Image dictionary (imgCIF) (A. P. HAMMERSLEY, H. J. BERNSTEIN AND J. D. WESTBROOK) 444
4.7. Symmetry dictionary (symCIF) (I. D. BROWN) 459
4.8. Molecular Information File dictionary (MIF) (F. H. ALLEN, J. M. BARNARD, A. P. F. COOK AND S. R. HALL) 467
4.9. DDL1 dictionary (S. R. HALL) 471
4.10. DDL2 dictionary (J. D. WESTBROOK AND S. R. HALL) 473

4.1. Core dictionary (coreCIF)

BY S. R. HALL, F. H. ALLEN AND I. D. BROWN

This is version 2.3.1 of the core CIF dictionary (coreCIF). A commentary on the use of this dictionary may be found in Chapter 3.2.

The data names defined here are central to the description and reporting of any crystal structure determination, and this dictionary collects the natural set of descriptors for small-unit-cell structures (typically inorganic or small-molecular organic and metal-organic compounds) determined in single-crystal experiments. These data items may be supplemented by additional items designed for use in powder-diffraction experiments (Chapter 3.3), modulated and composite structures (Chapter 3.4), or electron-density studies (Chapter 3.5).

The data items in this dictionary also form a suitable basis for the description of biological macromolecular structures, but the complexity of such structures requires a more extensive dictionary using a rigorously relational data model that expresses dependencies and inheritances between individual items. The macromolecular CIF (mmCIF) dictionary described in Chapter 3.6 and given in Chapter 4.5 provides a complete set of data items within this model, including the content of the core CIF dictionary in a modified formalism.

Categories are described in alphabetic order; data items are arranged alphabetically within each category.

ATOM_SITE

Data items in the ATOM_SITE category record details about the atom sites in a crystal structure, such as the positional coordinates, atomic displacement parameters, and magnetic moments and directions.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_adp_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
  O1   .4154(4)  .5699(1)  .3026(0)  .060(1)  Uani  ?    ?
  C2   .5630(5)  .5087(2)  .3246(1)  .060(2)  Uani  ?    ?
  C3   .5350(5)  .4920(2)  .3997(1)  .048(1)  Uani  ?    ?
  N4   .3570(3)  .5558(1)  .4167(0)  .039(1)  Uani  ?    ?
  C5   .3000(5)  .6122(2)  .3581(1)  .045(1)  Uani  ?    ?
  O21  .6958(5)  .4738(2)  .2874(1)  .090(2)  Uani  ?    ?
  C31  .4869(6)  .3929(2)  .4143(2)  .059(2)  Uani  ?    ?
 # - - - - data truncated for brevity - - - -
  H321C  .04(1)  .318(3)   .320(2)   .14000   Uiso  ?    ?
  H322A  .25(1)  .272(4)   .475(3)   .19000   Uiso  ?    ?
  H322B  .34976  .22118    .40954    .19000   Uiso  calc C322
  H322C  .08(1)  .234(4)   .397(3)   .19000   Uiso  ?    ?
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_atom_site_aniso_label
_atom_site_aniso_B_11
_atom_site_aniso_B_22
_atom_site_aniso_B_33
_atom_site_aniso_B_12
_atom_site_aniso_B_13
_atom_site_aniso_B_23
_atom_site_aniso_type_symbol
 O1   .071(1) .076(1) .0342(9) .008(1)   .0051(9) -.0030(9) O
 C2   .060(2) .072(2) .047(1)  .002(2)   .013(1)  -.009(1)  C
 C3   .038(1) .060(2) .044(1)  .007(1)   .001(1)  -.005(1)  C
 N4   .037(1) .048(1) .0325(9) .0025(9)  .0011(9) -.0011(9) N
 C5   .043(1) .060(1) .032(1)  .001(1)  -.001(1)   .001(1)  C
 # - - - - data truncated for brevity - - - -
 O21  .094(2) .109(2) .068(1)  .023(2)   .038(1)  -.010(1)  O
 C51  .048(2) .059(2) .049(1)  .002(1)  -.000(1)   .007(1)  C
 C511 .048(2) .071(2) .097(3) -.008(2)  -.003(2)   .010(2)  C
 C512 .078(2) .083(2) .075(2)  .009(2)  -.005(2)   .033(2)  C
 C513 .074(2) .055(2) .075(2)  .004(2)   .001(2)  -.010(2)  C
 # - - - - data truncated for brevity - - - -
```

*Example 3 – based on data set DPTD of Yamin, Suwandi, Fun, Sivakumar & bin Shawkataly [Acta Cryst. (1996), C**52**, 951–953].*

```
loop_
_atom_site_label
_atom_site_chemical_conn_number
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
 S1  1  0.74799(9)  -0.12482(11)  0.27574(9)  0.0742(3)
 S2  2  1.08535(10)  0.16131(9)   0.34061(9)  0.0741(3)
 N1  3  1.0650(2)   -0.1390(2)    0.2918(2)   0.0500(5)
 C1  4  0.9619(3)   -0.0522(3)    0.3009(2)   0.0509(6)
 # - - - - data truncated for brevity - - - -
```

Example 4 – hypothetical example to illustrate the description of a disordered methyl group.

```
loop_
  _atom_site_label     # *_assembly M is a disordered methyl
  _atom_site_occupancy # with configurations 'A' and 'B':
  _atom_site_disorder_assembly #
  _atom_site_disorder_group    #    H11B    H11A      H13B
                               #     .       |       .
  C1     1       .       .     #       .     |     .
  H11A   .5      M       A     #         .   |   .
  H12A   .5      M       A     #             C1 --------C2---
  H13A   .5      M       A     #           / .  \
  H11B   .5      M       B     #         /   .    \
  H12B   .5      M       B     #       /     .      \
  H13B   .5      M       B     #   H12A    H12B    H13A
```

_atom_site_adp_type (*char*)

A standard code used to describe the type of atomic displacement parameters used for the site.

Appears in list containing **_atom_site_label**.

Related item: **_atom_site_thermal_displace_type** (alternate).

The data value must be one of the following:

Uani	anisotropic U^{ij}
Uiso	isotropic U
Uovl	overall U
Umpe	multipole expansion U
Bani	anisotropic B^{ij}
Biso	isotropic B
Bovl	overall B

[atom_site]

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; FRANK H. ALLEN, Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge, CB2 1EZ, England; I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1.

_atom_site_aniso_B_11
_atom_site_aniso_B_12
_atom_site_aniso_B_13
_atom_site_aniso_B_22
_atom_site_aniso_B_23
_atom_site_aniso_B_33 (*numb, su*)

These are the standard anisotropic atomic displacement components in ångströms squared which appear in the structure-factor term

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Appears in list containing **_atom_site_aniso_label**.

Related item: **_atom_site_aniso_U_** (conversion). **[atom_site]**

_atom_site_aniso_label (*char*)

Anisotropic atomic displacement parameters are usually looped in a separate list. If this is the case, this code must match the **_atom_site_label** of the associated atom in the atom coordinate list and conform with the same rules described in **_atom_site_label**.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[atom_site]**

_atom_site_aniso_ratio (*numb*)

Ratio of the maximum to minimum principal axes of displacement (thermal) ellipsoids.

Appears in list containing **_atom_site_aniso_label**.

The permitted range is $1.0 \rightarrow \infty$. **[atom_site]**

_atom_site_aniso_type_symbol (*char*)

This **_atom_type_symbol** code links the anisotropic atom parameters to the atom-type data associated with this site and must match one of the **_atom_type_symbol** codes in this list.

Appears in list containing **_atom_site_aniso_label**. **Must** match parent data name **_atom_site_type_symbol**. **[atom_site]**

_atom_site_aniso_U_11
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_22
_atom_site_aniso_U_23
_atom_site_aniso_U_33 (*numb, su*)

These are the standard anisotropic atomic displacement components in ångströms squared which appear in the structure-factor term

$$T = \exp\Big\{-2\pi^2\sum_i\Big[\sum_j (U^{ij}h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

The unique elements of the real symmetric matrix are entered by row.

Appears in list containing **_atom_site_aniso_label**.

Related item: **_atom_site_aniso_B_** (conversion). **[atom_site]**

_atom_site_attached_hydrogens (*numb*)

The number of hydrogen atoms attached to the atom at this site excluding any hydrogen atoms for which coordinates (measured or calculated) are given.

Appears in list containing **_atom_site_label**.

The permitted range is $0 \rightarrow 8$. Where no value is given, the assumed value is '0'.

Examples: '2' (water oxygen), '1' (hydroxyl oxygen), '4' (ammonium nitrogen).

[atom_site]

_atom_site_B_equiv_geom_mean (*numb, su*)

Equivalent isotropic atomic displacement parameter, $B_{\rm equiv}$, in ångströms squared, calculated as the geometric mean of the anisotropic atomic displacement parameters.

$$B_{\rm equiv} = (B_i B_j B_k)^{1/3},$$

where B_n = the principal components of the orthogonalized B^{ij}.

The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Appears in list containing **_atom_site_label**.

The permitted range is $0.0 \rightarrow \infty$.

Related items:

_atom_site_B_iso_or_equiv (alternate),
_atom_site_U_equiv_geom_mean (conversion). **[atom_site]**

_atom_site_B_iso_or_equiv (*numb, su*)

Isotropic atomic displacement parameter, or equivalent isotropic atomic displacement parameter, $B_{\rm equiv}$, in ångströms squared, calculated from anisotropic displacement components.

$$B_{\rm equiv} = (1/3)\sum_i\Big[\sum_j (B^{ij}A_i A_j a_i^* a_j^*)\Big],$$

where A = the real-space cell lengths and a^* = the reciprocal-space cell lengths; $B^{ij} = 8\pi^2 U^{ij}$.

Reference: Fischer, R. X. & Tillmanns, E. (1988). *Acta Cryst.* C**44**, 775–776.

The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Appears in list containing **_atom_site_label**.

The permitted range is $0.0 \rightarrow \infty$.

Related items:

_atom_site_B_equiv_geom_mean (alternate),
_atom_site_U_iso_or_equiv (conversion). **[atom_site]**

_atom_site_calc_attached_atom (*char*)

The **_atom_site_label** of the atom site to which the 'geometry-calculated' atom site is attached.

Appears in list containing **_atom_site_label**. Where no value is given, the assumed value is '.'. **[atom_site]**

_atom_site_calc_flag (*char*)

A standard code to signal whether the site coordinates have been determined from the intensities or calculated from the geometry of surrounding sites, or have been assigned dummy coordinates. The abbreviation 'c' may be used in place of 'calc'.

Appears in list containing **_atom_site_label**.

The data value must be one of the following:

d	determined from diffraction measurements
calc	calculated from molecular geometry
c	abbreviation for 'calc'
dum	dummy site with meaningless coordinates

Where no value is given, the assumed value is 'd'. **[atom_site]**

_atom_site_Cartn_x
_atom_site_Cartn_y
_atom_site_Cartn_z (*numb, su*)

The atom-site coordinates in ångströms specified according to a set of orthogonal Cartesian axes related to the cell axes as specified by the **_atom_sites_Cartn_transform_axes** description.

Appears in list containing **_atom_site_label**.

Related item: **_atom_site_fract_** (alternate). **[atom_site]**

_atom_site_chemical_conn_number (*numb*)

This number links an atom site to the chemical connectivity list. It must match a number specified by **_chemical_conn_atom_number.**

Appears in list containing **_atom_site_label**. **Must** match parent data name **_chemical_conn_atom_number**.

The permitted range is $1 \rightarrow \infty$. **[atom_site]**

_atom_site_constraints (*char*)

A description of the constraints applied to parameters at this site during refinement. See also **_atom_site_refinement_flags** and **_refine_ls_number_constraints.**

Appears in list containing **_atom_site_label**. Where no value is given, the assumed value is '**.**'.

Example: '`pop=1.0-pop(Zn3)`'. **[atom_site]**

_atom_site_description (*char*)

A description of special aspects of this site. See also **_atom_site_refinement_flags.**

Appears in list containing **_atom_site_label**.

Example: '`Ag/Si disordered`'. **[atom_site]**

_atom_site_disorder_assembly (*char*)

A code which identifies a cluster of atoms that show long-range positional disorder but are locally ordered. Within each such cluster of atoms, **_atom_site_disorder_group** is used to identify the sites that are simultaneously occupied. This field is only needed if there is more than one cluster of disordered atoms showing independent local order.

Appears in list containing **_atom_site_label**.

Examples: '`A`' (disordered methyl assembly with groups 1 and 2), '`B`' (disordered sites related by a mirror), '`S`' (disordered sites independent of symmetry). **[atom_site]**

_atom_site_disorder_group (*char*)

A code which identifies a group of positionally disordered atom sites that are locally simultaneously occupied. Atoms that are positionally disordered over two or more sites (*e.g.* the hydrogen atoms of a methyl group that exists in two orientations) can be assigned to two or more groups. Sites belonging to the same group are simultaneously occupied, but those belonging to different groups are not. A minus prefix (*e.g.* '-1') is used to indicate sites disordered about a special position.

Appears in list containing **_atom_site_label**.

Examples: '`1`' (unique disordered site in group 1), '`2`' (unique disordered site in group 2), '`-1`' (symmetry-independent disordered site). **[atom_site]**

_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z (*numb, su*)

Atom-site coordinates as fractions of the **_cell_length_** values.

Appears in list containing **_atom_site_label**.

Related item: **_atom_site_Cartn_** (alternate). **[atom_site]**

_atom_site_label (*char*)

The **_atom_site_label** is a unique identifier for a particular site in the crystal. This code is made up of a sequence of up to seven components, **_atom_site_label_component_0** to ***_6**, which may be specified as separate data items. Component 0 usually matches one of the specified **_atom_type_symbol** codes. This is not mandatory if an **_atom_site_type_symbol** item is included in the atom-site list. The **_atom_site_type_symbol** always takes precedence over an **_atom_site_label** in the identification of the atom type. The label components 1 to 6 are optional, and normally only components 0 and 1 are used. Note that components 0 and 1 are concatenated, while all other components, if specified, are separated by an underscore. Underscore separators are only used if higher-order components exist. If an intermediate component is not used, it may be omitted provided the underscore separators are inserted. For example, the label 'C233__ggg' is acceptable and represents the components C, 233, '' and ggg. Different labels may have a different number of components.

Appears in list as essential element of loop structure. May match child data name(s):
_atom_site_aniso_label, **_geom_angle_atom_site_label_1**,
_geom_angle_atom_site_label_2, **_geom_angle_atom_site_label_3**,
_geom_bond_atom_site_label_1, **_geom_bond_atom_site_label_2**,
_geom_contact_atom_site_label_1,
_geom_contact_atom_site_label_2, **_geom_hbond_atom_site_label_D**,
_geom_hbond_atom_site_label_H, **_geom_hbond_atom_site_label_A**,
_geom_torsion_atom_site_label_1,
_geom_torsion_atom_site_label_2,
_geom_torsion_atom_site_label_3,
_geom_torsion_atom_site_label_4.

Examples: '`C12`', '`Ca3g28`', '`Fe3+17`', '`H*251`', '`boron2a`', '`C_a_phe_83_a_0`', '`Zn_Zn_301_A_0`'. **[atom_site]**

_atom_site_label_component_0
_atom_site_label_component_1
_atom_site_label_component_2
_atom_site_label_component_3
_atom_site_label_component_4
_atom_site_label_component_5
_atom_site_label_component_6 (*char*)

Component 0 is normally a code which matches identically with one of the **_atom_type_symbol** codes. If this is the case, then the rules governing the **_atom_type_symbol** code apply. If, however, the data item **_atom_site_type_symbol** is also specified in the atom-site list, component 0 need not match this symbol or adhere to any of the **_atom_type_symbol** rules. Component 1 is referred to as the 'atom number'. When component 0 is the atom-type code, it is used to number the sites with the same atom type. This component code must start with at least one digit which is not followed by a + or - sign (to distinguish it from the component 0 rules). Components 2 to 6 contain the identifier, residue, sequence, asymmetry identifier and alternate codes, respectively. These codes may be composed of any characters except an underscore.

Appears in list containing **_atom_site_label**. **[atom_site]**

_atom_site_occupancy (*numb, su*)

The fraction of the atom type present at this site. The sum of the occupancies of all the atom types at this site may not significantly exceed 1.0 unless it is a dummy site. The value must lie in the 99.97% Gaussian confidence interval $-3u \leq x \leq 1 + 3u$. The **_enumeration_range** of $0.0 \rightarrow 1.0$ is thus correctly interpreted as meaning $(0.0 - 3u) \leq x \leq (1.0 + 3u)$.

Appears in list containing **_atom_site_label**.

The permitted range is $0.0 \rightarrow 1.0$. Where no value is given, the assumed value is '`1.0`'. **[atom_site]**

_atom_site_refinement_flags (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _atom_site_refinement_flags_posn, _atom_site_refinement_flags_adp, _atom_site_refinement_flags_occupancy.

*A concatenated series of single-letter codes which indicate the refinement restraints or constraints applied to this site. This item should not be used. It has been replaced by _atom_site_refinement_flags_posn, *_adp and *_occupancy. It is retained in this dictionary only to provide compatibility with legacy CIFs.*

Appears in list containing _atom_site_label.

The data value must be one of the following:

.	*no refinement constraints*
S	*special-position constraint on site*
G	*rigid-group refinement of site*
R	*riding-atom site attached to non-riding atom*
D	*distance or angle restraint on site*
T	*thermal displacement constraints*
U	$U_{\rm iso}$ *or* U^{ij} *restraint (rigid bond)*
P	*partial occupancy constraint*

[atom_site]

_atom_site_refinement_flags_adp (*char*)

A code which indicates the refinement restraints or constraints applied to the atomic displacement parameters of this site.

Appears in list containing **_atom_site_label**.

Related item: **_atom_site_refinement_flags** (alternate).

The data value must be one of the following:

.	no constraints on atomic displacement parameters
T	special-position constraints on atomic displacement parameters
U	$U_{\rm iso}$ or U^{ij} restraint (rigid bond)
TU	both constraints applied

[atom_site]

_atom_site_refinement_flags_occupancy (*char*)

A code which indicates that refinement restraints or constraints were applied to the occupancy of this site.

Appears in list containing **_atom_site_label**.

Related item: **_atom_site_refinement_flags** (alternate).

The data value must be one of the following:

.	no constraints on site-occupancy parameters
P	site-occupancy constraint

[atom_site]

_atom_site_refinement_flags_posn (*char*)

A code which indicates the refinement restraints or constraints applied to the positional coordinates of this site.

Appears in list containing **_atom_site_label**.

Related item: **_atom_site_refinement_flags** (alternate).

The data value must be one of the following:

.	no constraints on positional coordinates
D	distance or angle restraint on positional coordinates
G	rigid-group refinement of positional coordinates
R	riding-atom site attached to non-riding atom
S	special-position constraint on positional coordinates
DG	combination of the above constraints
DR	combination of the above constraints
DS	combination of the above constraints
GR	combination of the above constraints
GS	combination of the above constraints
RS	combination of the above constraints
DGR	combination of the above constraints
DGS	combination of the above constraints
DRS	combination of the above constraints
GRS	combination of the above constraints
DGRS	combination of the above constraints

[atom_site]

_atom_site_restraints (*char*)

A description of restraints applied to specific parameters at this site during refinement. See also **_atom_site_refinement_flags** and **_refine_ls_number_restraints**.

Appears in list containing **_atom_site_label**.

Example: 'restrained to planar ring'. **[atom_site]**

_atom_site_symmetry_multiplicity (*numb*)

The multiplicity of a site due to the space-group symmetry as given in *International Tables for Crystallography* Vol. A (2002).

Appears in list containing **_atom_site_label**.

The permitted range is $1 \rightarrow 192$. **[atom_site]**

_atom_site_thermal_displace_type (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _atom_site_adp_type.

A standard code used to describe the type of atomic displacement parameters used for the site.

Appears in list containing _atom_site_label.

The data value must be one of the following:

Uani	*anisotropic* U^{ij}
Uiso	*isotropic U*
Uovl	*overall U*
Umpe	*multipole expansion U*
Bani	*anisotropic* B^{ij}
Biso	*isotropic B*
Bovl	*overall B*

[atom_site]

_atom_site_type_symbol (*char*)

A code to identify the atom species (singular or plural) occupying this site. This code must match a corresponding **_atom_type_symbol**. The specification of this code is optional if component 0 of the **_atom_site_label** is used for this purpose. See **_atom_type_symbol**.

Appears in list containing **_atom_site_label**. **Must** match parent data name **_atom_type_symbol**. May match child data name(s): **_atom_site_aniso_type_symbol**.

Examples: 'Cu', 'Cu2+', 'dummy', 'Fe3+Ni2+', 'S-', 'H*', 'H(SDS)'. **[atom_site]**

_atom_site_U_equiv_geom_mean (*numb, su*)

Equivalent isotropic atomic displacement parameter, $U_{\rm equiv}$, in ångströms squared, calculated as the geometric mean of the anisotropic atomic displacement parameters.

$$U_{\rm equiv} = (U_i U_j U_k)^{1/3},$$

where U_n = the principal components of the orthogonalized U^{ij}.

Appears in list containing **_atom_site_label**.

The permitted range is $0.0 \rightarrow \infty$.

Related items:

_atom_site_U_iso_or_equiv (alternate),
_atom_site_B_equiv_geom_mean (conversion). **[atom_site]**

_atom_site_U_iso_or_equiv (*numb, su*)

Isotropic atomic displacement parameter, or equivalent isotropic atomic displacement parameter, $U_{\rm equiv}$, in ångströms squared, calculated from anisotropic atomic displacement parameters.

$$U_{\rm equiv} = (1/3) \sum_i \left[\sum_j (U^{ij} A_i A_j a_i^* a_j^*) \right],$$

where A = the real-space cell lengths and a^* = the reciprocal-space cell lengths.

Reference: Fischer, R. X. & Tillmanns, E. (1988). *Acta Cryst.* C**44**, 775–776.

Appears in list containing **_atom_site_label**.

The permitted range is $0.0 \rightarrow \infty$.

Related items:

_atom_site_U_equiv_geom_mean (alternate),
_atom_site_B_iso_or_equiv (conversion). **[atom_site]**

_atom_site_Wyckoff_symbol (*char*)

The Wyckoff symbol (letter) as listed in the space-group tables of *International Tables for Crystallography* Vol. A (2002).

Appears in list containing **_atom_site_label**. **[atom_site]**

ATOM_SITES

Data items in the ATOM_SITES category record details about the crystallographic cell and cell transformations, which are common to all atom sites.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_atom_sites_Cartn_transform_axes
     'c along z, astar along x, b along y'

_atom_sites_Cartn_tran_matrix_11   58.39
_atom_sites_Cartn_tran_matrix_12    0.00
_atom_sites_Cartn_tran_matrix_13    0.00
_atom_sites_Cartn_tran_matrix_21    0.00
_atom_sites_Cartn_tran_matrix_22   86.70
_atom_sites_Cartn_tran_matrix_23    0.00
_atom_sites_Cartn_tran_matrix_31    0.00
_atom_sites_Cartn_tran_matrix_32    0.00
_atom_sites_Cartn_tran_matrix_33   46.27
```

_atom_sites_Cartn_tran_matrix_11
_atom_sites_Cartn_tran_matrix_12
_atom_sites_Cartn_tran_matrix_13
_atom_sites_Cartn_tran_matrix_21
_atom_sites_Cartn_tran_matrix_22
_atom_sites_Cartn_tran_matrix_23
_atom_sites_Cartn_tran_matrix_31
_atom_sites_Cartn_tran_matrix_32
_atom_sites_Cartn_tran_matrix_33 (*numb*)

Matrix elements used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates. The axial alignments of this transformation are described in **_atom_sites_Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites_Cartn_tran_vector_**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites_Cartn_tran_vector_1
_atom_sites_Cartn_tran_vector_2
_atom_sites_Cartn_tran_vector_3 (*numb*)

Elements of a 3 × 1 translation vector used in the transformation of fractional coordinates in the ATOM_SITE category to Cartesian coordinates. The axial alignments of this transformation are described in **_atom_sites_Cartn_transform_axes**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites_Cartn_transform_axes (*char*)

A description of the relative alignment of the crystal cell axes to the Cartesian orthogonal axes as applied in the transformation matrix **_atom_sites_Cartn_tran_matrix_**.

Example: 'a parallel to x; b in the plane of y and z'.

[atom_sites]

_atom_sites_fract_tran_matrix_11
_atom_sites_fract_tran_matrix_12
_atom_sites_fract_tran_matrix_13
_atom_sites_fract_tran_matrix_21
_atom_sites_fract_tran_matrix_22
_atom_sites_fract_tran_matrix_23
_atom_sites_fract_tran_matrix_31
_atom_sites_fract_tran_matrix_32
_atom_sites_fract_tran_matrix_33 (*numb*)

Matrix elements used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates. The axial alignments of this transformation are described in **_atom_sites_Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites_fract_tran_vector_**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites_fract_tran_vector_1
_atom_sites_fract_tran_vector_2
_atom_sites_fract_tran_vector_3 (*numb*)

Elements of a 3 × 1 translation vector used in the transformation of Cartesian coordinates in the ATOM_SITE category to fractional coordinates. The axial alignments of this transformation are described in **_atom_sites_Cartn_transform_axes**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites_solution_primary
_atom_sites_solution_secondary
_atom_sites_solution_hydrogens (*char*)

Codes which identify the methods used to locate the initial atom sites. The ***_primary** code identifies how the first atom sites were determined; the ***_secondary** code identifies how the remaining non-hydrogen sites were located; and the ***_hydrogens** code identifies how the hydrogen sites were located.

The data value must be one of the following:

difmap	difference Fourier map
vecmap	real-space vector search
heavy	heavy-atom method
direct	structure-invariant direct methods
geom	inferred from neighbouring sites
disper	anomalous-dispersion techniques
isomor	isomorphous structure methods

[atom_sites]

_atom_sites_special_details (*char*)

Additional information about the atomic coordinates not coded elsewhere in the CIF.

[atom_sites]

ATOM_TYPE

Data items in the ATOM_TYPE category record details about properties of the atoms that occupy the atom sites, such as the atomic scattering factors.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_scat_dispersion_real
_atom_type_scat_dispersion_imag
_atom_type_scat_source
  C 0 72  .017  .009  International_Tables_Vol_IV_Table_2.2B
  H 0 100  0     0    International_Tables_Vol_IV_Table_2.2B
  O 0 12  .047  .032  International_Tables_Vol_IV_Table_2.2B
  N 0 4   .029  .018  International_Tables_Vol_IV_Table_2.2B
```

_atom_type_analytical_mass_% (*numb*)
Mass percentage of this atom type derived from chemical analysis.
Appears in list containing **_atom_type_symbol**.
The permitted range is 0.0 → 100.0. **[atom_type]**

_atom_type_description (*char*)
A description of the atom(s) designated by this atom type. In most cases, this will be the element name and oxidation state of a single atom species. For disordered or nonstoichiometric structures it will describe a combination of atom species.
Appears in list containing **_atom_type_symbol**.
Examples: 'deuterium', '0.34Fe+0.66Ni'. **[atom_type]**

_atom_type_number_in_cell (*numb*)
Total number of atoms of this atom type in the unit cell.
Appears in list containing **_atom_type_symbol**.
The permitted range is 0 → ∞. **[atom_type]**

_atom_type_oxidation_number (*numb*)
Formal oxidation state of this atom type in the structure.
Appears in list containing **_atom_type_symbol**.
The permitted range is −8 → 8. Where no value is given, the assumed value is '0'.
[atom_type]

_atom_type_radius_bond
_atom_type_radius_contact (*numb*)
The effective intra- and intermolecular bonding radii in ångströms of this atom type.
Appears in list containing **_atom_type_symbol**.
The permitted range is 0.0 → 5.0. **[atom_type]**

_atom_type_scat_Cromer_Mann_a1
_atom_type_scat_Cromer_Mann_a2
_atom_type_scat_Cromer_Mann_a3
_atom_type_scat_Cromer_Mann_a4
_atom_type_scat_Cromer_Mann_b1
_atom_type_scat_Cromer_Mann_b2
_atom_type_scat_Cromer_Mann_b3
_atom_type_scat_Cromer_Mann_b4
_atom_type_scat_Cromer_Mann_c (*numb*)
The Cromer–Mann scattering-factor coefficients used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.
Appears in list containing **_atom_type_symbol**. **[atom_type]**

_atom_type_scat_dispersion_imag
_atom_type_scat_dispersion_real (*numb*)
The imaginary and real components of the anomalous-dispersion scattering factor, f'' and f', in electrons for this atom type and the radiation given in **_diffrn_radiation_wavelength**.
Appears in list containing **_atom_type_symbol**. Where no value is given, the assumed value is '0.0'. **[atom_type]**

_atom_type_scat_dispersion_source (*char*)
Reference to source of real and imaginary dispersion corrections for scattering factors used for this atom type.
Appears in list containing **_atom_type_symbol**.
Example: 'International Tables Vol. IV Table 2.3.1'. **[atom_type]**

_atom_type_scat_length_neutron (*numb*)
The bound coherent scattering length in femtometres for the atom type at the isotopic composition used for the diffraction experiment.
Appears in list containing **_atom_type_symbol**. Where no value is given, the assumed value is '0.0'. **[atom_type]**

_atom_type_scat_source (*char*)
Reference to source of scattering factors or scattering lengths used for this atom type.
Appears in list containing **_atom_type_symbol**.
Example: 'International Tables Vol. IV Table 2.4.6B'. **[atom_type]**

_atom_type_scat_versus_stol_list (*char*)
A table of scattering factors as a function of $(\sin\theta)/\lambda$. This table should be well commented to indicate the items present. Regularly formatted lists are strongly recommended.
Appears in list containing **_atom_type_symbol**. **[atom_type]**

_atom_type_symbol (*char*)
The code used to identify the atom species (singular or plural) representing this atom type. Normally this code is the element symbol. The code may be composed of any character except an underscore with the additional proviso that digits designate an oxidation state and must be followed by a + or - character.
Appears in list as essential element of loop structure. May match child data name(s): **_atom_site_type_symbol**.
Examples: 'C', 'Cu2+', 'H(SDS)', 'dummy', 'FeNi'. **[atom_type]**

AUDIT

Data items in the AUDIT category record details about the creation and subsequent updating of the data block.

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_audit_block_code                 TOZ_1991-03-20

_audit_creation_date              1991-03-20
_audit_creation_method  from_xtal_archive_file_using_CIFIO
_audit_update_record
; 1991-04-09     text and data added by Tony Willis.
  1991-04-15     rec'd by co-editor as manuscript HL0007.
  1991-04-17     adjustments based on first referee report.
  1991-04-18     adjustments based on second referee report.
;
```

_audit_block_code (*char*)
A code intended to identify uniquely the current data block.
Example: 'TOZ_1991-03-20'. **[audit]**

_audit_creation_date (*char*)
The date that the data block was created. The date format is *yyyy-mm-dd*.
Example: '1990-07-12'. **[audit]**

_audit_creation_method (*char*)
A description of how data were entered into the data block.
Example: 'spawned by the program QBEE'. **[audit]**

_audit_update_record (*char*)
A record of any changes to the data block. The update format is a date (*yyyy-mm-dd*) followed by a description of the changes. The latest update entry is added to the bottom of this record.
Example: '1990-07-15 Updated by the Co-editor'. **[audit]**

AUDIT_AUTHOR

Data items in the AUDIT_AUTHOR category record details about the author(s) of the data block.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_audit_author_name
_audit_author_address
    'Fitzgerald, Paula M. D.'
;  Department of Biophysical Chemistry
   Merck Research Laboratories
   PO Box 2000, Ry80M203
   Rahway
   New Jersey 07065
   USA
;
    'Van Middlesworth, J. F.'
;  Department of Biophysical Chemistry
   Merck Research Laboratories
   PO Box 2000, Ry80M203
   Rahway
   New Jersey 07065
   USA
;
```

_audit_author_address (*char*)

The address of an author of this data block. If there are multiple authors, **_audit_author_address** is looped with **_audit_author_name**.

Appears in list containing **_audit_author_name**.

Example:

```
; Department
  Institute
  Street
  City and postcode
  COUNTRY
;
```

[audit_author]

_audit_author_name (*char*)

The name of an author of this data block. If there are multiple authors, **_audit_author_name** is looped with **_audit_author_address**. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).

Appears in list as essential element of loop structure.

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.', 'M\"uller, H.A.', 'Ross II, C.R.'.

[audit_author]

AUDIT_CONFORM

Data items in the AUDIT_CONFORM category describe the dictionary versions against which the data names appearing in the current data block are conformant.

Example 1 – any file conforming to the current CIF core dictionary.

```
_audit_conform_dict_name        cif_core.dic
_audit_conform_dict_version     2.3.1
_audit_conform_dict_location
      ftp://ftp.iucr.org/pub/cif_core.2.3.1.dic
```

_audit_conform_dict_location (*char*)

A file name or uniform resource locator (URL) for the dictionary to which the current data block conforms.

May appear in list containing **_audit_conform_dict_name**. **[audit_conform]**

_audit_conform_dict_name (*char*)

The string identifying the highest-level dictionary defining data names used in this file.

May appear in list as essential element of loop structure. **[audit_conform]**

_audit_conform_dict_version (*char*)

The version number of the dictionary to which the current data block conforms.

May appear in list containing **_audit_conform_dict_name**. **[audit_conform]**

AUDIT_CONTACT_AUTHOR

Data items in the AUDIT_CONTACT_AUTHOR category record details about the name and address of the author to be contacted concerning the contents of this data block.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_audit_contact_author_name
_audit_contact_author_address
_audit_contact_author_email
_audit_contact_author_fax
_audit_contact_author_phone
    'Fitzgerald, Paula M. D.'
;  Department of Biophysical Chemistry
   Merck Research Laboratories
   PO Box 2000, Ry80M203
   Rahway
   New Jersey 07065
   USA
;
    'paula_fitzgerald@merck.com'
    '1(908)5945510'
    '1(908)5945510'
```

_audit_contact_author_address (*char*)

The mailing address of the author of the data block to whom correspondence should be addressed.

Example:

```
; Department
  Institute
  Street
  City and postcode
  COUNTRY
;
```

[audit_contact_author]

_audit_contact_author_email (*char*)

The electronic mail address of the author of the data block to whom correspondence should be addressed, in a form recognizable to international networks. The format of e-mail addresses is given in Section 3.4, *Address Specification*, of *Internet Message Format*, RFC 2822, P. Resnick (Editor), Network Standards Group, April 2001.

Examples: 'name@host.domain.country', 'bm@iucr.org'.

[audit_contact_author]

_audit_contact_author_fax (*char*)

The facsimile telephone number of the author of the data block to whom correspondence should be addressed. The recommended style starts with the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces.

Examples: '12(34)9477334', '12()349477334'. **[audit_contact_author]**

_audit_contact_author_name (*char*)

The name of the author of the data block to whom correspondence should be addressed. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.', 'M\"uller, H.A.', 'Ross II, C.R.'.

[audit_contact_author]

_audit_contact_author_phone (*char*)
The telephone number of the author of the data block to whom correspondence should be addressed. The recommended style starts with the international dialing prefix, followed by the area code in parentheses, followed by the local number and any extension number prefixed by 'x', with no spaces.
Examples: '12(34)9477330', '12()349477330', '12(34)9477330x5543'.
[audit_contact_author]

AUDIT_LINK

Data items in the AUDIT_LINK category record details about the relationships between data blocks in the current CIF.

Example 1 – multiple structure paper, as illustrated in A Guide to CIF for Authors (1995). IUCr: Chester.

```
loop_
_audit_link_block_code
_audit_link_block_description
   .         'discursive text of paper with two structures'
   morA_(1)  'structure 1 of 2'
   morA_(2)  'structure 2 of 2'
```

Example 2 – example file for the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
_audit_link_block_code
_audit_link_block_description
  .       'publication details'
  KSE_COM 'experimental data common to ref./mod. structures'
  KSE_REF 'reference structure'
  KSE_MOD 'modulated structure'
```

_audit_link_block_code (*char*)
The value of **_audit_block_code** associated with a data block in the current file related to the current data block. The special value '.' may be used to refer to the current data block for completeness.
Appears in list as essential element of loop structure. [audit_link]

_audit_link_block_description (*char*)
A textual description of the relationship of the referenced data block to the current one.
Appears in list containing **_audit_link_block_code**. [audit_link]

CELL

Data items in the CELL category record details about the crystallographic cell parameters and their measurement.

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_cell_length_a                  5.959(1)
_cell_length_b                  14.956(1)
_cell_length_c                  19.737(3)
_cell_angle_alpha               90
_cell_angle_beta                90
_cell_angle_gamma               90
_cell_volume                    1759.0(3)

_cell_measurement_temperature   293
_cell_measurement_reflns_used   25
_cell_measurement_theta_min     25
_cell_measurement_theta_max     31
```

_cell_angle_alpha
_cell_angle_beta
_cell_angle_gamma (*numb, su*)
Unit-cell angles of the reported structure in degrees. The values of **_refln_index_h**, ***_k**, ***_l** must correspond to the cell defined by these values and **_cell_length_a**, ***_b** and ***_c**. The values of **_diffrn_refln_index_h**, ***_k**, ***_l** may not correspond to these values if a cell transformation took place following the measurement of the diffraction intensities. See also **_diffrn_reflns_transf_matrix_**.
The permitted range is 0.0 → 180.0. Where no value is given, the assumed value is '90.0'.
[cell]

_cell_formula_units_Z (*numb*)
The number of the formula units in the unit cell as specified by **_chemical_formula_structural**, **_chemical_formula_moiety** or **_chemical_formula_sum**.
The permitted range is 1 → ∞. [cell]

_cell_length_a
_cell_length_b
_cell_length_c (*numb, su*)
Unit-cell lengths in ångströms corresponding to the structure reported. The values of **_refln_index_h**, ***_k**, ***_l** must correspond to the cell defined by these values and **_cell_angle_** values. The values of **_diffrn_refln_index_h**, ***_k**, ***_l** may not correspond to these values if a cell transformation took place following the measurement of the diffraction intensities. See also **_diffrn_reflns_transf_matrix_**.
The permitted range is 0.0 → ∞. [cell]

_cell_measurement_pressure (*numb, su*)
The pressure in kilopascals at which the unit-cell parameters were measured (not the pressure at which the sample was synthesized).
The permitted range is 0.0 → ∞. [cell]

_cell_measurement_radiation (*char*)
Description of the radiation used to measure the unit-cell data. See also **_cell_measurement_wavelength**.
Examples: 'neutron', 'Cu K\a', 'synchrotron'. [cell]

_cell_measurement_reflns_used (*numb*)
The total number of reflections used to determine the unit cell. These reflections may be specified as **_cell_measurement_refln_** data items.
[cell]

_cell_measurement_temperature (*numb, su*)
The temperature in kelvins at which the unit-cell parameters were measured (not the temperature of synthesis).
The permitted range is 0.0 → ∞. [cell]

_cell_measurement_theta_max
_cell_measurement_theta_min (*numb*)
The maximum and minimum θ angles of reflections used to measure the unit cell in degrees.
The permitted range is 0.0 → 90.0. [cell]

_cell_measurement_wavelength (*numb*)
The wavelength in ångströms of the radiation used to measure the unit cell. If this is not specified, the wavelength is assumed to be the same as that given in **_diffrn_radiation_wavelength**.
The permitted range is 0.0 → ∞. [cell]

_cell_reciprocal_angle_alpha
_cell_reciprocal_angle_beta
_cell_reciprocal_angle_gamma (*numb, su*)

The angles defining the reciprocal cell in degrees. These are related to those in the real cell by

$$\cos\alpha^* = (\cos\beta\cos\gamma - \cos\alpha)/(\sin\beta\sin\gamma),$$
$$\cos\beta^* = (\cos\gamma\cos\alpha - \cos\beta)/(\sin\gamma\sin\alpha),$$
$$\cos\gamma^* = (\cos\alpha\cos\beta - \cos\gamma)/(\sin\alpha\sin\beta).$$

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.

The permitted range is 0.0 → 180.0. Where no value is given, the assumed value is '90.0'. **[cell]**

_cell_reciprocal_length_a
_cell_reciprocal_length_b
_cell_reciprocal_length_c (*numb, su*)

The reciprocal-cell lengths in inverse ångströms. These are related to the real cell by

$$a^* = bc\sin\alpha/V,$$
$$b^* = ca\sin\beta/V,$$
$$c^* = ab\sin\gamma/V,$$

where V is the cell volume.

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.

The permitted range is 0.0 → ∞. **[cell]**

_cell_special_details (*char*)

A description of special aspects of the cell choice, noting possible alternative settings.

Examples: 'pseudo-orthorhombic',
'standard setting from 45 deg rotation around c'. **[cell]**

_cell_volume (*numb, su*)

Cell volume V in ångströms cubed.

$$V = abc(1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma)^{1/2},$$

where a = **_cell_length_a**, b = **_cell_length_b**, c = **_cell_length_c**, α = **_cell_angle_alpha**, β = **_cell_angle_beta** and γ = **_cell_angle_gamma**.

The permitted range is 0.0 → ∞. **[cell]**

CELL_MEASUREMENT_REFLN

Data items in the CELL_MEASUREMENT_REFLN category record details about the reflections used in the determination of the crystallographic cell parameters. The **_cell_measurement_refln_** data items would in general be used only for diffractometer measurements.

Example 1 – extracted from the CAD-4 listing for $Rb_2S_2O_6$ at room temperature (unpublished).

```
loop_
 _cell_measurement_refln_index_h
 _cell_measurement_refln_index_k
 _cell_measurement_refln_index_l
 _cell_measurement_refln_theta
   -2    4    1          8.67
    0    3    2          9.45
    3    0    2          9.46
   -3    4    1          8.93
   -2    1   -2          7.53
   10    0    0         23.77
    0   10    0         23.78
   -5    4    1         11.14
  # - - - - data truncated for brevity - - - -
```

_cell_measurement_refln_index_h
_cell_measurement_refln_index_k
_cell_measurement_refln_index_l (*numb*)

Miller indices of a reflection used for measurement of the unit cell.

Appears in list as essential element of loop structure. **[cell_measurement_refln]**

_cell_measurement_refln_theta (*numb*)

θ angle in degrees for the reflection used for measurement of the unit cell with the indices **_cell_measurement_refln_index_**.

Appears in list containing **_cell_measurement_refln_index_**.

The permitted range is 0.0 → 90.0. **[cell_measurement_refln]**

CHEMICAL

Data items in the CHEMICAL category record details about the composition and chemical properties of the compound. The formula data items must agree with those that specify the density, unit-cell and Z values.

*Example 1 – based on data set 9597gaus of Alyea, Ferguson & Kannan [Acta Cryst. (1996), **C52**, 765–767].*

```
_chemical_name_systematic
  trans-bis(tricyclohexylphosphine)tetracarbonylmolybdenum(0)
```

_chemical_absolute_configuration (*char*)

Necessary conditions for the assignment of **_chemical_absolute_configuration** are given by H. D. Flack and G. Bernardinelli (1999, 2000).

References: Flack, H. D. & Bernardinelli, G. (1999). *Acta Cryst.* A**55**, 908–915; Flack, H. D. & Bernardinelli, G. (2000). *J. Appl. Cryst.* **33**, 1143–1148.

The data value must be one of the following:

rm	Absolute configuration established by the structure determination of a compound containing a chiral reference molecule of known absolute configuration.
ad	Absolute configuration established by anomalous-dispersion effects in diffraction measurements on the crystal.
rmad	Absolute configuration established by the structure determination of a compound containing a chiral reference molecule of known absolute configuration and confirmed by anomalous-dispersion effects in diffraction measurements on the crystal.
syn	Absolute configuration has not been established by anomalous-dispersion effects in diffraction measurements on the crystal. The enantiomer has been assigned by reference to an unchanging chiral centre in the synthetic procedure.
unk	Absolute configuration is unknown, there being no firm chemical evidence for its assignment to hand and it having not been established by anomalous-dispersion effects in diffraction measurements on the crystal. An arbitrary choice of enantiomer has been made.
.	Inapplicable.

[chemical]

_chemical_compound_source (*char*)

Description of the source of the compound under study, or of the parent molecule if a simple derivative is studied. This includes the place of discovery for minerals or the actual source of a natural product.

Examples: 'From Norilsk (USSR)',
'Extracted from the bark of Cinchona Naturalis'. **[chemical]**

_chemical_melting_point (*numb, su*)

The temperature in kelvins at which the crystalline solid changes to a liquid.

The permitted range is 0.0 → ∞. **[chemical]**

_chemical_melting_point_gt
_chemical_melting_point_lt (*numb*)
A temperature in kelvins below which (*_lt) or above which (*_gt) the melting point (the temperature at which the crystalline solid changes to a liquid) lies. These items allow a range of temperatures to be given. **_chemical_melting_point** should always be used in preference to these items whenever possible.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_chemical_melting_point** (alternate). **[chemical]**

_chemical_name_common (*char*)
Trivial name by which the compound is commonly known.
Example: '1-bromoestradiol'. **[chemical]**

_chemical_name_mineral (*char*)
Mineral name accepted by the International Mineralogical Association. Use only for natural minerals. See also **_chemical_compound_source**.
Example: 'chalcopyrite'. **[chemical]**

_chemical_name_structure_type (*char*)
Commonly used structure-type name. Usually only applied to minerals or inorganic compounds.
Examples: 'perovskite', 'sphalerite', 'A15'. **[chemical]**

_chemical_name_systematic (*char*)
IUPAC or *Chemical Abstracts* full name of the compound.
Example: '1-bromoestra-1,3,5(10)-triene-3,17\b-diol'. **[chemical]**

_chemical_optical_rotation (*char*)
The optical rotation in solution of the compound is specified in the following format:

$$[\alpha]^{\mathrm{TEMP}}_{\mathrm{WAVE}} = \mathrm{SORT} \quad (c = \mathrm{CONC},\ \mathrm{SOLV}),$$

where TEMP is the temperature of the measurement in degrees Celsius, WAVE is an indication of the wavelength of the light used for the measurement, CONC is the concentration of the solution given as the mass of the substance in g per 100 ml of solution, SORT is the signed value (preceded by a + or a - sign) of $100\alpha/(lc)$, where α is the signed optical rotation in degrees measured in a cell of length l in dm and c is the value of *CONC* as defined above, and SOLV is the chemical formula of the solvent.
Example: '[\a]^25^~D~ = +108 (c = 3.42, CHCl~3~)'. **[chemical]**

_chemical_properties_biological (*char*)
A free-text description of the biological properties of the material.
Examples:

```
; diverse biological activities including use as a
  laxative and strong antibacterial activity against
  S. aureus and weak activity against
  cyclooxygenase-1 (COX-1)
;
; antibiotic activity against Bacillus subtilis
  (ATCC 6051) but no significant activity against
  Candida albicans (ATCC 14053), Aspergillus flavus
  (NRRL 6541) and Fusarium verticillioides (NRRL
  25457)
;
; weakly potent lipoxygenase nonredox inhibitor
;
; no influenza A virus sialidase inhibitory and
  plaque reduction activities
;
; low toxicity against Drosophila melanogaster
;
```
[chemical]

_chemical_properties_physical (*char*)
A free-text description of the physical properties of the material.
Examples: 'air-sensitive', 'moisture-sensitive', 'hygroscopic', 'deliquescent', 'oxygen-sensitive', 'photo-sensitive', 'pyrophoric', 'semiconductor', 'ferromagnetic at low temperature', 'paramagnetic and thermochromic'. **[chemical]**

_chemical_temperature_decomposition (*numb, su*)
The temperature in kelvins at which the solid decomposes.
The permitted range is $0.0 \rightarrow \infty$.
Example: '350'. **[chemical]**

_chemical_temperature_decomposition_gt
_chemical_temperature_decomposition_lt (*numb*)
A temperature in kelvins below which (*_lt) or above which (*_gt) the solid is known to decompose. These items allow a range of temperatures to be given. **_chemical_temperature_decomposition** should always be used in preference to these items whenever possible.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_chemical_temperature_decomposition** (alternate).
Example: '350'. **[chemical]**

_chemical_temperature_sublimation (*numb, su*)
The temperature in kelvins at which the solid sublimes.
The permitted range is $0.0 \rightarrow \infty$.
Example: '350'. **[chemical]**

_chemical_temperature_sublimation_gt
_chemical_temperature_sublimation_lt (*numb*)
A temperature in kelvins below which (*_lt) or above which (*_gt) the solid is known to sublime. These items allow a range of temperatures to be given. **_chemical_temperature_sublimation** should always be used in preference to these items whenever possible.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_chemical_temperature_sublimation** (alternate).
Example: '350'. **[chemical]**

CHEMICAL_CONN_ATOM

Data items in the **_chemical_conn_atom_** and **_chemical_conn_bond_** categories record details about the two-dimensional (2D) chemical structure of the molecular species. They allow a 2D chemical diagram to be reconstructed for use in a publication or in a database search for structural and substructural relationships. The **_chemical_conn_atom_** data items provide information about the chemical properties of the atoms in the structure. In cases where crystallographic and molecular symmetry elements coincide, they must also contain symmetry-generated atoms, so that the **_chemical_conn_atom_** and **_chemical_conn_bond_** data items will always describe a complete chemical entity.

*Example 1 – based on data set DPTD of Yamin, Suwandi, Fun, Sivakumar & bin Shawkataly [Acta Cryst. (1996), C**52**, 951–953].*

```
loop_
_chemical_conn_atom_number
_chemical_conn_atom_type_symbol
_chemical_conn_atom_display_x
_chemical_conn_atom_display_y
_chemical_conn_atom_NCA
_chemical_conn_atom_NH
    1   S    .39  .81   1   0
    2   S    .39  .96   2   0
    3   N    .14  .88   3   0
    4   C    .33  .88   3   0
    5   C    .11  .96   2   2
    6   C    .03  .96   2   2
    7   C    .03  .80   2   2
    8   C    .11  .80   2   2
    9   S    .54  .81   1   0
    10  S    .54  .96   2   0
    11  N    .80  .88   3   0
    12  C    .60  .88   3   0
    13  C    .84  .96   2   2
    14  C    .91  .96   2   2
    15  C    .91  .80   2   2
    16  C    .84  .80   2   2
```

_chemical_conn_atom_charge (*numb*)

The net integer charge assigned to this atom. This is the formal charge assignment normally found in chemical diagrams.

Appears in list containing **_chemical_conn_atom_type_symbol**.

The permitted range is $-6 \rightarrow 6$. Where no value is given, the assumed value is '0'.

Examples: '1' (for an ammonium nitrogen), '-1' (for a chloride ion).

[chemical_conn_atom]

_chemical_conn_atom_display_x
_chemical_conn_atom_display_y (*numb*)

The 2D Cartesian coordinates (x, y) of the position of this atom in a recognizable chemical diagram. The coordinate origin is at the lower left corner, the x axis is horizontal and the y axis is vertical. The coordinates must lie in the range 0.0 to 1.0. These coordinates can be obtained from projections of a suitable uncluttered view of the molecular structure.

Appears in list containing **_chemical_conn_atom_type_symbol**.

The permitted range is $0.0 \rightarrow 1.0$. **[chemical_conn_atom]**

_chemical_conn_atom_NCA (*numb*)

The number of connected atoms excluding terminal hydrogen atoms.

Appears in list containing **_chemical_conn_atom_type_symbol**.

The permitted range is $0 \rightarrow \infty$. **[chemical_conn_atom]**

_chemical_conn_atom_NH (*numb*)

The total number of hydrogen atoms attached to this atom, regardless of whether they are included in the refinement or the **_atom_site_** list. This number will be the same as **_atom_site_attached_hydrogens** only if none of the hydrogen atoms appear in the **_atom_site_** list.

Appears in list containing **_chemical_conn_atom_type_symbol**.

The permitted range is $0 \rightarrow \infty$. **[chemical_conn_atom]**

_chemical_conn_atom_number (*numb*)

The chemical sequence number to be associated with this atom.

Appears in list containing **_chemical_conn_atom_type_symbol**. May match child data name(s): **_atom_site_chemical_conn_number**, **_chemical_conn_bond_atom_1**, **_chemical_conn_bond_atom_2**.

The permitted range is $1 \rightarrow \infty$. **[chemical_conn_atom]**

_chemical_conn_atom_type_symbol (*char*)

A code identifying the atom type. This code must match an **_atom_type_symbol** code in the **_atom_type_** list or be a recognizable element symbol.

Appears in list as essential element of loop structure. **[chemical_conn_atom]**

CHEMICAL_CONN_BOND

Data items in the **_chemical_conn_atom_** and **_chemical_conn_bond_** categories record details about the two-dimensional (2D) chemical structure of the molecular species. They allow a 2D chemical diagram to be reconstructed for use in a publication or in a database search for structural and substructural relationships. The **_chemical_conn_bond_** data items specify the connections between the atoms in the **_chemical_conn_atom_** list and the nature of the chemical bond between these atoms.

*Example 1 – based on data set DPTD of Yamin, Suwandi, Fun, Sivakumar & bin Shawkataly [Acta Cryst. (1996), C**52**, 951–953].*

```
loop_
_chemical_conn_bond_atom_1
_chemical_conn_bond_atom_2
_chemical_conn_bond_type
   4     1     doub     4     3     sing
   4     2     sing     5     3     sing
   6     5     sing     7     6     sing
   8     7     sing     8     3     sing
   10    2     sing     12    9     doub
   12    11    sing     12    10    sing
   13    11    sing     14    13    sing
   15    14    sing     16    15    sing
   16    11    sing     17    5     sing
   18    5     sing     19    6     sing
   20    6     sing     21    7     sing
   22    7     sing     23    8     sing
   24    8     sing     25    13    sing
   26    13    sing     27    14    sing
   28    14    sing     29    15    sing
   30    15    sing     31    16    sing
   32    16    sing
```

_chemical_conn_bond_atom_1
_chemical_conn_bond_atom_2 (*numb*)

Atom numbers which must match with chemical sequence numbers specified as **_chemical_conn_atom_number** values. These link the bond connection to the chemical numbering and atom sites.

Appears in list. **Must** match parent data name **_chemical_conn_atom_number**.

The permitted range is $1 \rightarrow \infty$. **[chemical_conn_bond]**

_chemical_conn_bond_type (*char*)

The chemical bond type associated with the connection between the two sites **_chemical_conn_bond_atom_1** and ***_2**.

Appears in list containing **_chemical_conn_bond_atom_**.

The data value must be one of the following:

sing	single bond
doub	double bond
trip	triple bond
quad	quadruple bond
arom	aromatic bond
poly	polymeric bond
delo	delocalized double bond
pi	π bond

Where no value is given, the assumed value is 'sing'. **[chemical_conn_bond]**

CHEMICAL_FORMULA

_chemical_formula_ items specify the composition and chemical properties of the compound. The formula data items must agree with those that specify the density, unit-cell and *Z* values. The following rules apply to the construction of the data items **_chemical_formula_analytical**, ***_structural** and ***_sum**. For the data item ***_moiety**, the formula construction is broken up into residues or moieties, *i.e.* groups of atoms that form a molecular unit or molecular ion. The rules given below apply within each moiety but different requirements apply to the way that moieties are connected (see **_chemical_formula_moiety**). (1) Only recognized element symbols may be used. (2) Each element symbol is followed by a 'count' number. A count of '1' may be omitted. (3) A space or parenthesis must separate each cluster of (element symbol + count). (4) Where a group of elements is enclosed in parentheses, the multiplier for the group must follow the closing parenthesis. That is, all element and group multipliers are assumed to be printed as subscripted numbers. (An exception to this rule exists for ***_moiety** formulae where pre- and post-multipliers are permitted for molecular units.) (5) Unless the elements are ordered in a manner that corresponds to their chemical structure, as in **_chemical_formula_structural**, the order of the elements within any group or moiety depends on whether carbon is present or not. If carbon is present, the order should be: C, then H, then the other elements in alphabetical order of their symbol. If carbon is not present, the elements are listed purely in alphabetical order of their symbol. This is the 'Hill' system used by *Chemical Abstracts*. This ordering is used in **_chemical_formula_moiety** and **_chemical_formula_sum**.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_chemical_formula_moiety          'C18 H25 N O3'
_chemical_formula_sum             'C18 H25 N O3'
_chemical_formula_weight          303.40
```

*Example 2 – based on data set 9597gaus of Alyea, Ferguson & Kannan [Acta Cryst. (1996), C**52**, 765–767].*

```
_chemical_formula_iupac      '[Mo (C O)4 (C18 H33 P)2]'
_chemical_formula_moiety     'Mo,4(C O),2(C18 H33 P)'
_chemical_formula_structural '((C O)4 (P (C6 H11)3)2)Mo'
_chemical_formula_sum         'C40 H66 Mo O4 P2'
_chemical_formula_weight      768.81
```

_chemical_formula_analytical (*char*)
Formula determined by standard chemical analysis including trace elements. See the CHEMICAL_FORMULA category description for rules for writing chemical formulae. Parentheses are used only for standard uncertainties (e.s.d.'s).
Example: 'Fe2.45(2) Ni1.60(3) S4'. **[chemical_formula]**

_chemical_formula_iupac (*char*)
Formula expressed in conformance with IUPAC rules for inorganic and metal-organic compounds where these conflict with the rules for any other **_chemical_formula_** entries. Typically used for formatting a formula in accordance with journal rules. This should appear in the data block in addition to the most appropriate of the other **_chemical_formula_** data names.

Reference: IUPAC (1990). *Nomenclature of Inorganic Chemistry*. Oxford: Blackwell Scientific Publications.
Example: '[Co Re (C12 H22 P)2 (C O)6].0.5C H3 O H'. **[chemical_formula]**

_chemical_formula_moiety (*char*)
Formula with each discrete bonded residue or ion shown as a separate moiety. See the CHEMICAL_FORMULA category description for rules for writing chemical formulae. In addition to the general formulae requirements, the following rules apply: (1) Moieties are separated by commas ',' . (2) The order of elements within a moiety follows general rule (5) in the CHEMICAL_FORMULA category description. (3) Parentheses are not used within moieties but may surround a moiety. Parentheses may not be nested. (4) Charges should be placed at the end of the moiety. The charge '+' or '-' may be preceded by a numerical multiplier and should be separated from the last (element symbol + count) by a space. Pre- or post-multipliers may be used for individual moieties.
Examples: 'C7 H4 Cl Hg N O3 S', 'C12 H17 N4 O S 1+, C6 H2 N3 O7 1-', 'C12 H16 N2 O6, 5(H2 O1)', '(Cd 2+)3, (C6 N6 Cr 3-)2, 2(H2 O)'. **[chemical_formula]**

_chemical_formula_structural (*char*)
See the CHEMICAL_FORMULA category description for the rules for writing chemical formulae for inorganics, organometallics, metal complexes *etc.*, in which bonded groups are preserved as discrete entities within parentheses, with post-multipliers as required. The order of the elements should give as much information as possible about the chemical structure. Parentheses may be used and nested as required. This formula should correspond to the structure as actually reported, *i.e.* trace elements not included in atom-type and atom-site lists should not be included in this formula (see also **_chemical_formula_analytical**).
Examples: 'Ca ((Cl O3)2 O)2 (H2 O)6', '(Pt (N H3)2 (C5 H7 N3 O)2) (Cl O4)2'. **[chemical_formula]**

_chemical_formula_sum (*char*)
See the CHEMICAL_FORMULA category description for the rules for writing chemical formulae in which all discrete bonded residues and ions are summed over the constituent elements, following the ordering given in general rule (5) in the CHEMICAL_FORMULA category description. Parentheses are not normally used.
Example: 'C18 H19 N7 O8 S'. **[chemical_formula]**

_chemical_formula_weight (*numb*)
Formula mass in daltons. This mass should correspond to the formulae given under **_chemical_formula_structural**, ***_iupac**, ***_moiety** or ***_sum** and, together with the *Z* value and cell parameters, should yield the density given as **_exptl_crystal_density_diffrn**.
The permitted range is $1.0 \rightarrow \infty$. **[chemical_formula]**

_chemical_formula_weight_meas (*numb*)
Formula mass in daltons measured by a non-diffraction experiment.
The permitted range is $1.0 \rightarrow \infty$. **[chemical_formula]**

CITATION

Data items in the CITATION category record details about the literature cited as being relevant to the contents of the data block.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_citation_id
_citation_coordinate_linkage
_citation_title
_citation_country
_citation_page_first
_citation_page_last
_citation_year
_citation_journal_abbrev
_citation_journal_volume
_citation_journal_issue
_citation_journal_id_ASTM
_citation_journal_id_ISSN
_citation_book_title
_citation_book_publisher
_citation_book_id_ISBN
_citation_special_details
  primary  yes
; Crystallographic analysis of a complex between human
  immunodeficiency virus type 1 protease and
  acetyl-pepstatin at 2.0-Angstroms resolution.
;
  US  14209  14219  1990  'J. Biol. Chem.'  265  .
  HBCHA3  0021-9258  .  .  .
; The publication that directly relates to this coordinate
  set.
;
  2  no
; Three-dimensional structure of aspartyl-protease from
  human immunodeficiency virus HIV-1.
;
  UK  615  619  1989  'Nature'  337  .
  NATUAS  0028-0836  .  .  .
; Determination of the structure of the unliganded enzyme.
;
  3 no
; Crystallization of the aspartylprotease from human
  immunodeficiency virus, HIV-1.
;
  US  1919  1921  1989  'J. Biol. Chem.'  264  .
  HBCHA3  0021-9258  .  .  .
; Crystallization of the unliganded enzyme.
;
```

_citation_abstract (*char*)
Abstract for the citation. This is used most when the citation is extracted from a bibliographic database that contains full text or abstract information.
Appears in list containing **_citation_id**. **[citation]**

_citation_abstract_id_CAS (*char*)
The *Chemical Abstracts* Service (CAS) abstract identifier; relevant for journal articles.
Appears in list containing **_citation_id**. **[citation]**

_citation_book_id_ISBN (*char*)
The International Standard Book Number (ISBN) code assigned to the book cited; relevant for books or book chapters.
Appears in list containing **_citation_id**. **[citation]**

_citation_book_publisher (*char*)
The name of the publisher of the citation; relevant for books or book chapters.
Appears in list containing **_citation_id**.
Example: 'John Wiley'. **[citation]**

_citation_book_publisher_city (*char*)
The location of the publisher of the citation; relevant for books or book chapters.
Appears in list containing **_citation_id**.
Example: 'New York'. **[citation]**

_citation_book_title (*char*)
The title of the book in which the citation appeared; relevant for books or book chapters.
Appears in list containing **_citation_id**. **[citation]**

_citation_coordinate_linkage (*char*)
_citation_coordinate_linkage states whether or not this citation is concerned with precisely the set of coordinates given in the data block. If, for instance, the publication described the same structure, but the coordinates had undergone further refinement prior to creation of the data block, the value of this data item would be 'no'.
Appears in list containing **_citation_id**.
The data value must be one of the following:

no	citation unrelated to current coordinates
n	abbreviation for 'no'
yes	citation related to current coordinates
y	abbreviation for 'yes'

[citation]

_citation_country (*char*)
The country of publication; relevant for books and book chapters.
Appears in list containing **_citation_id**. **[citation]**

_citation_database_id_CSD (*char*)
Identifier ('refcode') of the database record in the Cambridge Structural Database that contains details of the cited structure.
Appears in list containing **_citation_id**.
Example: 'LEKKUH'. **[citation]**

_citation_database_id_Medline (*numb*)
Accession number used by Medline to categorize a specific bibliographic entry.
Appears in list containing **_citation_id**.
The permitted range is $1 \rightarrow \infty$.
Example: '89064067'. **[citation]**

_citation_id (*char*)
The value of **_citation_id** must uniquely identify a record in the **_citation_** list. The **_citation_id** 'primary' should be used to indicate the citation that the author(s) consider to be the most pertinent to the contents of the data block. Note that this item need not be a number; it can be any unique identifier.
Appears in list as essential element of loop structure. May match child data name(s): **_citation_author_citation_id**, **_citation_editor_citation_id**.
Examples: 'primary', '1', '2', '3'. **[citation]**

_citation_journal_abbrev (*char*)
Abbreviated name of the journal cited as given in the *Chemical Abstracts* Service *Source Index*.
Appears in list containing **_citation_id**.
Example: 'J. Mol. Biol.'. **[citation]**

_citation_journal_full (*char*)
Full name of the journal cited; relevant for journal articles.
Appears in list containing **_citation_id**.
Example: 'Journal of Molecular Biology'. **[citation]**

_citation_journal_id_ASTM (*char*)
The American Society for Testing and Materials (ASTM) code assigned to the journal cited (also referred to as the CODEN designator of the *Chemical Abstracts* Service); relevant for journal articles.
Appears in list containing **_citation_id**. **[citation]**

_citation_journal_id_CSD (*char*)
The Cambridge Structural Database (CSD) code assigned to the journal cited; relevant for journal articles. This is also the system used at the Protein Data Bank (PDB).
Appears in list containing **_citation_id**.
Example: '0070'. **[citation]**

_citation_journal_id_ISSN (*char*)
The International Standard Serial Number (ISSN) code assigned to the journal cited; relevant for journal articles.
Appears in list containing **_citation_id**. **[citation]**

_citation_journal_issue (*char*)
Issue number of the journal cited; relevant for journal articles.
Appears in list containing **_citation_id**.
Example: '2'. **[citation]**

_citation_journal_volume (*char*)
Volume number of the journal cited; relevant for journal articles.
Appears in list containing **_citation_id**.
Example: '174'. **[citation]**

_citation_language (*char*)
Language in which the cited article is written.
Appears in list containing **_citation_id**.
Example: 'German'. **[citation]**

_citation_page_first
_citation_page_last (*char*)
The first and last pages of the citation; relevant for journal articles, books and book chapters.
Appears in list containing **_citation_id**. **[citation]**

_citation_special_details (*char*)
A description of special aspects of the relationship of the contents of the data block to the literature item cited.
Appears in list containing **_citation_id**.
Examples:
```
; citation relates to this precise coordinate set
;
; citation relates to earlier low-resolution structure
;
; citation relates to further refinement of structure
  reported in citation 2
;
```
[citation]

_citation_title (*char*)
The title of the citation; relevant for journal articles, books and book chapters.
Appears in list containing **_citation_id**.
Example:
```
; Structure of diferric duck ovotransferrin at 2.35 \%A
  resolution.
;
```
[citation]

_citation_year (*numb*)
The year of the citation; relevant for journal articles, books and book chapters.
Appears in list containing **_citation_id**.
Example: '1984'. **[citation]**

CITATION_AUTHOR

Data items in the CITATION_AUTHOR category record details about the authors associated with the citations in the **_citation_** list.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_citation_author_citation_id
_citation_author_name
  primary  'Fitzgerald, P.M.D.'
  primary  'McKeever, B.M.'
  primary  'Van Middlesworth, J.F.'
  primary  'Springer, J.P.'
  primary  'Heimbach, J.C.'
  primary  'Leu, C.-T.'
  primary  'Herber, W.K.'
  primary  'Dixon, R.A.F.'
  primary  'Darke, P.L.'
  2        'Navia, M.A.'
  2        'Fitzgerald, P.M.D.'
  2        'McKeever, B.M.'
  2        'Leu, C.-T.'
  2        'Heimbach, J.C.'
  2        'Herber, W.K.'
  2        'Sigal, I.S.'
  2        'Darke, P.L.'
  2        'Springer, J.P.'
  3        'McKeever, B.M.'
  3        'Navia, M.A.'
  3        'Fitzgerald, P.M.D.'
  3        'Springer, J.P.'
  3        'Leu, C.-T.'
  3        'Heimbach, J.C.'
  3        'Herber, W.K.'
  3        'Sigal, I.S.'
  3        'Darke, P.L.'
```

_citation_author_citation_id (*char*)
The value of **_citation_author_citation_id** must match an identifier specified by **_citation_id** in the **_citation_** list.
Appears in list as essential element of loop structure. **Must** match parent data name **_citation_id**. **[citation_author]**

_citation_author_name (*char*)
Name of an author of the citation; relevant for journal articles, books and book chapters. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).
Appears in list as essential element of loop structure.
Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.', 'M\"uller, H.A.', 'Ross II, C.R.'.
[citation_author]

_citation_author_ordinal (*char*)
This data name defines the order of the author's name in the list of authors of a citation.
Appears in list. **[citation_author]**

CITATION_EDITOR

Data items in the CITATION_EDITOR category record details about the editor associated with the book or book chapter citations in the `_citation_` list.

Example 1 – hypothetical example.

```
loop_
_citation_editor_citation_id
_citation_editor_name
  5        'McKeever, B.M.'
  5        'Navia, M.A.'
  5        'Fitzgerald, P.M.D.'
  5        'Springer, J.P.'
```

_citation_editor_citation_id (*char*)
The value of `_citation_editor_citation_id` must match an identifier specified by `_citation_id` in the `_citation_` list.
Appears in list as essential element of loop structure. **Must** match parent data name `_citation_id`. **[citation_editor]**

_citation_editor_name (*char*)
Name of an editor of the citation; relevant for books and book chapters. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).
Appears in list as essential element of loop structure.
Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.', 'M\"uller, H.A.', 'Ross II, C.R.'.
[citation_editor]

_citation_editor_ordinal (*char*)
This data name defines the order of the editor's name in the list of editors of a citation.
Appears in list. **[citation_editor]**

COMPUTING

Data items in the COMPUTING category record details about the computer programs used in the crystal structure analysis.

*Example 1 – Rodrìguez-Romero, Ruiz-Pérez & Solans [Acta Cryst. (1996), **C52**, 1415–1417].*

```
_computing_data_collection      'CAD-4 (Enraf-Nonius, 1989)'
_computing_cell_refinement      'CAD-4 (Enraf-Nonius, 1989)'
_computing_data_reduction       'CFEO (Solans, 1978)'
_computing_structure_solution   'SHELXS86 (Sheldrick, 1990)'
_computing_structure_refinement 'SHELXL93 (Sheldrick, 1993)'
_computing_molecular_graphics   'ORTEPII (Johnson, 1976)'
_computing_publication_material 'PARST (Nardelli, 1983)'
```

_computing_cell_refinement
_computing_data_collection
_computing_data_reduction
_computing_molecular_graphics
_computing_publication_material
_computing_structure_refinement
_computing_structure_solution (*char*)
Software used in the processing of the data. Give the program or package name and a brief reference.
Examples: 'CAD-4 (Enraf-Nonius, 1989)',
'DIFDAT, SORTRF, ADDREF (Hall & Stewart, 1990)',
'FRODO (Jones, 1986), ORTEP (Johnson, 1965)',
'CRYSTALS (Watkin, 1988)', 'SHELX85 (Sheldrick, 1985)'.
[computing]

DATABASE

Data items in the DATABASE category record details about the database identifiers of the data block. These data items are assigned by database managers and should only appear in a CIF if they originate from that source.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_database_code_CSD                'VOBYUG'
```

_database_code_CAS
_database_code_CSD
_database_code_ICSD
_database_code_MDF
_database_code_NBS
_database_code_PDB
_database_code_PDF (*char*)
The codes are assigned by databases: *Chemical Abstracts*; Cambridge Structural Database (organic and metal-organic compounds); Inorganic Crystal Structure Database; Metals Data File (metal structures); NBS (NIST) Crystal Data Database (lattice parameters); Protein Data Bank; and the Powder Diffraction File (JCPDS/ICDD).
[database]

_database_code_depnum_ccdc_archive (*char*)
Deposition numbers assigned by the Cambridge Crystallographic Data Centre (CCDC) to files containing structural information archived by the CCDC.
[database]

_database_code_depnum_ccdc_fiz (*char*)
Deposition numbers assigned by the Fachinformationszentrum Karlsruhe (FIZ) to files containing structural information archived by the Cambridge Crystallographic Data Centre (CCDC).
[database]

_database_code_depnum_ccdc_journal (*char*)
Deposition numbers assigned by various journals to files containing structural information archived by the Cambridge Crystallographic Data Centre (CCDC).
[database]

_database_CSD_history (*char*)
A history of changes made by the Cambridge Crystallographic Data Centre and incorporated into the Cambridge Structural Database (CSD).
[database]

_database_journal_ASTM
_database_journal_CSD (*char*)
The ASTM CODEN designator for a journal as given in the *Chemical Source List* maintained by the *Chemical Abstracts* Service, and the journal code used in the Cambridge Structural Database.
[database]

DIFFRN

Data items in the DIFFRN category record details about the intensity measurements.

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_diffrn_special_details
; \q scan width (1.0 + 0.14tan\q)\%, \q scan rate
  1.2\% min^-1^. Background counts for 5 s on each side
  every scan.
;

_diffrn_ambient_temperature        293
```

_diffrn_ambient_environment (*char*)
The gas or liquid surrounding the sample, if not air.
Examples: 'He', 'vacuum', 'mother liquor'. **[diffrn]**

_diffrn_ambient_pressure (*numb, su*)
The mean hydrostatic pressure in kilopascals at which the intensities were measured.
The permitted range is 0.0 → ∞. **[diffrn]**

_diffrn_ambient_pressure_gt
_diffrn_ambient_pressure_lt (*numb*)
The mean hydrostatic pressure in kilopascals above which (*_gt) or below which (*_lt) the intensities were measured. These items allow for a pressure range to be given. **_diffrn_ambient_pressure** should always be used in preference to these items whenever possible.
The permitted range is 0.0 → ∞.
Related item: **_diffrn_ambient_pressure** (alternate). **[diffrn]**

_diffrn_ambient_temperature (*numb, su*)
The mean temperature in kelvins at which the intensities were measured.
The permitted range is 0.0 → ∞. **[diffrn]**

_diffrn_ambient_temperature_gt
_diffrn_ambient_temperature_lt (*numb*)
The mean temperature in kelvins above which (*_gt) or below which (*_lt) the intensities were measured. These items allow a range of temperatures to be given. **_diffrn_ambient_temperature** should always be used in preference to these items whenever possible.
The permitted range is 0.0 → ∞.
Related item: **_diffrn_ambient_temperature** (alternate). **[diffrn]**

_diffrn_crystal_treatment (*char*)
Remarks about how the crystal was treated prior to the intensity measurements. Particularly relevant when intensities were measured at low temperature.
Examples: 'equilibrated in hutch for 24 hours',
'flash frozen in liquid nitrogen',
'slow cooled with direct air stream'. **[diffrn]**

_diffrn_measured_fraction_theta_full (*numb*)
Fraction of unique (symmetry-independent) reflections measured out to **_diffrn_reflns_theta_full**.
The permitted range is 0 → 1.0. **[diffrn]**

_diffrn_measured_fraction_theta_max (*numb*)
Fraction of unique (symmetry-independent) reflections measured out to **_diffrn_reflns_theta_max**.
The permitted range is 0 → 1.0. **[diffrn]**

_diffrn_special_details (*char*)
Special details of the intensity-measurement process. Should include information about source instability, crystal motion, degradation and so on.
Example:

```
; The results may not be entirely reliable
  as the measurement was made during a heat
  wave when the air-conditioning had failed.
;
```

[diffrn]

_diffrn_symmetry_description (*char*)
Observed diffraction point symmetry, systematic absences and possible space group(s) or superspace group(s) compatible with these.

[diffrn]

DIFFRN_ATTENUATOR

Data items in the DIFFRN_ATTENUATOR category record details about the diffraction attenuator scales employed.

Example 1 – hypothetical example.

```
loop_
   _diffrn_attenuator_code
   _diffrn_attenuator_scale
      0         1.00
      1        16.97
      2        33.89
```

_diffrn_attenuator_code (*char*)
A code associated with a particular attenuator setting. This code is referenced by the **_diffrn_refln_attenuator_code** which is stored with the intensities. See **_diffrn_attenuator_scale**.
Appears in list as essential element of loop structure. May match child data name(s):
_diffrn_refln_attenuator_code. **[diffrn_attenuator]**

_diffrn_attenuator_material (*char*)
Material from which the attenuator is made.
Appears in list containing **_diffrn_attenuator_code**. **[diffrn_attenuator]**

_diffrn_attenuator_scale (*numb*)
The scale factor applied when an intensity measurement is reduced by an attenuator identified by **_diffrn_attenuator_code**. The measured intensity must be multiplied by this scale to convert it to the same scale as unattenuated intensities.
Appears in list containing **_diffrn_attenuator_code**.
The permitted range is 1.0 → ∞. **[diffrn_attenuator]**

DIFFRN_DETECTOR

Data items in the DIFFRN_DETECTOR category describe the detector used to measure the scattered radiation, including any analyser and post-sample collimation.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_detector                  'multiwire'
_diffrn_detector_type             'Siemens'
```

_diffrn_detector (*char*)
The general class of the radiation detector.
Related item: **_diffrn_radiation_detector** (alternate).
Examples: 'photographic film', 'scintillation counter', 'CCD plate',
'BF~3~ counter'. **[diffrn_detector]**

_diffrn_detector_area_resol_mean (*numb*)
The resolution of an area detector, in pixels mm^{-1}.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_detector]**

_diffrn_detector_details (*char*)
A description of special aspects of the radiation detector.
[diffrn_detector]

_diffrn_detector_dtime (*numb*)
The deadtime in microseconds of the detector used to measure the diffraction intensities.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_diffrn_radiation_detector_dtime** (alternate).
[diffrn_detector]

_diffrn_detector_type (*char*)
The make, model or name of the detector device used.
[diffrn_detector]

_diffrn_radiation_detector (*char*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_detector.
The detector used to measure the diffraction intensities.
[diffrn_detector]

_diffrn_radiation_detector_dtime (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_detector_dtime.
The deadtime in microseconds of the detector used to measure the diffraction intensities.
The permitted range is $0.0 \rightarrow \infty$. ***[diffrn_detector]***

DIFFRN_MEASUREMENT

Data items in the DIFFRN_MEASUREMENT category refer to the mounting of the sample and to the goniometer on which it is mounted.

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_diffrn_measurement_device_type
          'Philips PW1100/20 diffractometer'
_diffrn_measurement_method         \q/2\q
```

_diffrn_measurement_details (*char*)
A description of special aspects of the intensity measurement.
Example: '440 frames of 0.25\%'. **[diffrn_measurement]**

_diffrn_measurement_device (*char*)
The general class of goniometer or device used to support and orient the specimen.
Examples: 'three-circle diffractometer',
'four-circle diffractometer', '\k-geometry diffractometer',
'oscillation camera', 'precession camera'. **[diffrn_measurement]**

_diffrn_measurement_device_details (*char*)
A description of special aspects of the device used to measure the diffraction intensities.
Example:
```
; commercial goniometer modified locally to
  allow for 90\% \t arc
;
```
[diffrn_measurement]

_diffrn_measurement_device_type (*char*)
The make, model or name of the measurement device (goniometer) used.
[diffrn_measurement]

_diffrn_measurement_method (*char*)
Method used to measure the intensities.
Example: 'profile data from \q/2\q scans'. **[diffrn_measurement]**

_diffrn_measurement_specimen_support (*char*)
The physical device used to support the crystal during data collection.
Examples: 'glass capillary', 'quartz capillary', 'fiber', 'metal loop'.
[diffrn_measurement]

DIFFRN_ORIENT_MATRIX

Data items in the DIFFRN_ORIENT_MATRIX category record details about the orientation matrix used in the measurement of the diffraction intensities.

Example 1 – data set n-alkylation_C-4 of Hussain, Fleming, Norman & Chang [Acta Cryst. (1996), C52, 1010–1012].

```
_diffrn_orient_matrix_UB_11          -0.04170
_diffrn_orient_matrix_UB_12          -0.01429
_diffrn_orient_matrix_UB_13          -0.02226
_diffrn_orient_matrix_UB_21          -0.00380
_diffrn_orient_matrix_UB_22          -0.05578
_diffrn_orient_matrix_UB_23          -0.05048
_diffrn_orient_matrix_UB_31           0.00587
_diffrn_orient_matrix_UB_32          -0.13766
_diffrn_orient_matrix_UB_33           0.02277

_diffrn_orient_matrix_type 'TEXSAN convention (MSC, 1989)'
```

_diffrn_orient_matrix_type (*char*)
A description of the orientation matrix type and how it should be applied to define the orientation of the crystal precisely with respect to the diffractometer axes.
[diffrn_orient_matrix]

_diffrn_orient_matrix_UB_11
_diffrn_orient_matrix_UB_12
_diffrn_orient_matrix_UB_13
_diffrn_orient_matrix_UB_21
_diffrn_orient_matrix_UB_22
_diffrn_orient_matrix_UB_23
_diffrn_orient_matrix_UB_31
_diffrn_orient_matrix_UB_32
_diffrn_orient_matrix_UB_33 (*numb*)
The elements of the diffractometer orientation matrix. These define the dimensions of the reciprocal cell and its orientation to the local diffractometer axes. See **_diffrn_orient_matrix_type**.
[diffrn_orient_matrix]

DIFFRN_ORIENT_REFLN

Data items in the DIFFRN_ORIENT_REFLN category record details about the reflections that define the orientation matrix used in the measurement of the diffraction intensities.

Example 1 – typical output listing from an Enraf-Nonius CAD-4 diffractometer.

```
loop_
_diffrn_orient_refln_index_h
_diffrn_orient_refln_index_k
_diffrn_orient_refln_index_l
_diffrn_orient_refln_angle_theta
_diffrn_orient_refln_angle_phi
_diffrn_orient_refln_angle_omega
_diffrn_orient_refln_angle_kappa
  -3   2   3    7.35   44.74   2.62    17.53
  -4   1   0    9.26   83.27   8.06     5.79
   0   0   6    5.85  -43.93 -25.36    86.20
   2   1   3    7.36  -57.87   6.26     5.42
   0   0  -6    5.85 -161.59  36.96   -86.79
  -3   1   0    6.74   80.28   5.87     2.60
   2   0   3    5.86  -76.86  -0.17    21.34
   0   0  12   11.78  -44.02 -19.51    86.41
   0   0 -12   11.78 -161.67  42.81   -86.61
  -5   1   0   11.75   86.24   9.16     7.44
   0   4   6   11.82  -19.82  10.45     4.19
   5   0   6   14.13  -77.28  10.17    15.34
   8   0   0   20.79  -77.08  25.30   -13.96
```

_diffrn_orient_refln_angle_chi
_diffrn_orient_refln_angle_kappa
_diffrn_orient_refln_angle_omega
_diffrn_orient_refln_angle_phi
_diffrn_orient_refln_angle_psi
_diffrn_orient_refln_angle_theta (*numb*)

Diffractometer angles of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix_UB_** and **_diffrn_orient_refln_index_h**, *_k and *_l.

Appears in list containing **_diffrn_orient_refln_index_**.

[diffrn_orient_refln]

_diffrn_orient_refln_index_h
_diffrn_orient_refln_index_k
_diffrn_orient_refln_index_l (*numb*)

The indices of a reflection used to define the orientation matrix. See **_diffrn_orient_matrix_**.

Appears in list as essential element of loop structure. **[diffrn_orient_refln]**

DIFFRN_RADIATION

Data items in the DIFFRN_RADIATION category describe the radiation used in measuring the diffraction intensities, its collimation and monochromatization before the sample. Post-sample treatment of the beam is described by data items in the DIFFRN_DETECTOR category.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_diffrn_radiation_type            'Cu K\a'
_diffrn_radiation_monochromator   'graphite'
```

_diffrn_radiation_collimation (*char*)

The collimation or focusing applied to the radiation.

Examples: '`0.3 mm double-pinhole`', '`0.5 mm`', '`focusing mirrors`'.

[diffrn_radiation]

_diffrn_radiation_filter_edge (*numb*)

Absorption edge in ångströms of the radiation filter used.

The permitted range is $0.0 \rightarrow \infty$. **[diffrn_radiation]**

_diffrn_radiation_inhomogeneity (*numb*)

Half-width in millimetres of the incident beam in the direction perpendicular to the diffraction plane.

The permitted range is $0.0 \rightarrow \infty$. **[diffrn_radiation]**

_diffrn_radiation_monochromator (*char*)

The method used to obtain monochromatic radiation. If a monochromator crystal is used, the material and the indices of the Bragg reflection are specified.

Examples: '`Zr filter`', '`Ge 220`', '`none`', '`equatorial mounted graphite`'.

[diffrn_radiation]

_diffrn_radiation_polarisn_norm (*numb*)

The angle in degrees, as viewed from the specimen, between the perpendicular component of the polarization and the diffraction plane. See **_diffrn_radiation_polarisn_ratio**.

The permitted range is $-180.0 \rightarrow 180.0$. **[diffrn_radiation]**

_diffrn_radiation_polarisn_ratio (*numb*)

Polarization ratio of the diffraction beam incident on the crystal. It is the ratio of the perpendicularly polarized to the parallel polarized components of the radiation. The perpendicular component forms an angle of **_diffrn_radiation_polarisn_norm** to the normal to the diffraction plane of the sample (*i.e.* the plane containing the incident and reflected beams).

The permitted range is $0.0 \rightarrow \infty$. **[diffrn_radiation]**

_diffrn_radiation_probe (*char*)

The nature of the radiation used (*i.e.* the name of the subatomic particle or the region of the electromagnetic spectrum). It is strongly recommended that this information be given, so that the probe radiation can be simply determined.

The data value must be one of the following:

`x-ray`
`neutron`
`electron`
`gamma`

[diffrn_radiation]

_diffrn_radiation_type (*char*)

The type of the radiation. This is used to give a more detailed description than **_diffrn_radiation_probe** and is typically a description of the X-ray wavelength in Siegbahn notation.

Examples: '`Cu K\a`', '`Cu K\a~1~`', '`Cu K-L~2,3~`', '`white-beam`'.

[diffrn_radiation]

_diffrn_radiation_xray_symbol (*char*)

The IUPAC symbol for the X-ray wavelength for the probe radiation.

The data value must be one of the following:

`K-L~3~`	$K\alpha_1$ in older Siegbahn notation
`K-L~2~`	$K\alpha_2$ in older Siegbahn notation
`K-M~3~`	$K\beta$ in older Siegbahn notation
`K-L~2,3~`	use where *K*-L_3 and *K*-L_2 are not resolved

[diffrn_radiation]

DIFFRN_RADIATION_WAVELENGTH

Data items in the DIFFRN_RADIATION_WAVELENGTH category describe the wavelength of the radiation used in measuring the diffraction intensities. Items may be looped to identify and assign weights to distinct wavelength components from a polychromatic beam.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_diffrn_radiation_wavelength      1.5418
```

_diffrn_radiation_wavelength (*numb*)
The radiation wavelength in ångströms.
May appear in list containing **_diffrn_radiation_wavelength_id**.
The permitted range is 0.0 → ∞. **[diffrn_radiation_wavelength]**

_diffrn_radiation_wavelength_id (*char*)
An arbitrary code identifying each value of **_diffrn_radiation_wavelength**. Items in the DIFFRN_RADIATION category are looped when multiple wavelengths are used. This code is used to link with the **_diffrn_refln_** list. It must match with one of the **_diffrn_refln_wavelength_id** codes.
Appears in list as essential element of loop structure. May match child data name(s): **_diffrn_refln_wavelength_id**.
Examples: 'x1', 'x2', 'neut'. **[diffrn_radiation_wavelength]**

_diffrn_radiation_wavelength_wt (*numb*)
The relative weight of a wavelength identified by the code **_diffrn_radiation_wavelength_id** in the list of wavelengths.
Appears in list containing **_diffrn_radiation_wavelength_id**.
The permitted range is 0.0 → 1.0. Where no value is given, the assumed value is '1.0'.
[diffrn_radiation_wavelength]

DIFFRN_REFLN

Data items in the DIFFRN_REFLN category record details about the intensities measured in the diffraction experiment. The DIFFRN_REFLN data items refer to individual intensity measurements and must be included in looped lists.

(The DIFFRN_REFLNS data items specify the parameters that apply to all intensity measurements. The DIFFRN_REFLNS data items are not looped.)

Example 1 – extracted from the CAD-4 listing for $Tl_2Cd_2(SO_4)_3$ at 85 K (unpublished).

```
loop_
     _diffrn_refln_index_h
     _diffrn_refln_index_k
     _diffrn_refln_index_l
     _diffrn_refln_angle_chi
     _diffrn_refln_scan_rate
     _diffrn_refln_counts_bg_1
     _diffrn_refln_counts_total
     _diffrn_refln_counts_bg_2
     _diffrn_refln_angle_theta
     _diffrn_refln_angle_phi
     _diffrn_refln_angle_omega
     _diffrn_refln_angle_kappa
     _diffrn_refln_scan_width
     _diffrn_refln_elapsed_time

 0  0 -16  0. 4.12  28 127  36  33.157  -75.846  16.404
50.170  1.516 19.43
 0  0 -15  0. 4.12  38 143  28  30.847  -75.846  14.094
50.170  1.516 19.82
 0  0 -14  0. 1.03 142 742 130  28.592  -75.846  11.839
50.170  1.516 21.32
 0  0 -13  0. 4.12  26 120  37  26.384  -75.846   9.631
50.170  1.450 21.68
 0  0 -12  0. 0.97 129 618 153  24.218  -75.846   7.464
50.170  1.450 23.20
 0  0 -11  0. 4.12  33 107  38  22.087  -75.846   5.334
50.170  1.384 23.55
 0  0 -10  0. 4.12  37 146  33  19.989  -75.846   3.235
50.170  1.384 23.90
 0  0  -9  0. 4.12  50 179  49  17.918  -75.846   1.164
50.170  1.384 24.25
# - - - - data truncated for brevity - - - -
 3  4  -4  0. 1.03  69 459  73  30.726  -53.744  46.543
-47.552  1.516 2082.58
 3  4  -5  0. 1.03  91 465  75  31.407  -54.811  45.519
-42.705  1.516 2084.07
 3 14  -6  0. 1.03  84 560  79  32.228  -55.841  44.745
-38.092  1.516 2085.57
# - - - - data truncated for brevity - - - -
```

_diffrn_refln_angle_chi
_diffrn_refln_angle_kappa
_diffrn_refln_angle_omega
_diffrn_refln_angle_phi
_diffrn_refln_angle_psi
_diffrn_refln_angle_theta (*numb*)
The diffractometer angles of a reflection in degrees. These correspond to the specified orientation matrix and the original measured cell before any subsequent cell transformations.
Appears in list containing **_diffrn_refln_index_**. **[diffrn_refln]**

_diffrn_refln_attenuator_code (*char*)
The code identifying the attenuator setting for this reflection. This code must match one of the **_diffrn_attenuator_code** values.
Appears in list containing **_diffrn_refln_index_**. **Must** match parent data name **_diffrn_attenuator_code**. **[diffrn_refln]**

_diffrn_refln_class_code (*char*)
The code identifying the class to which this reflection has been assigned. This code must match a value of **_diffrn_reflns_class_code**. Reflections may be grouped into classes for a variety of purposes. For example, for modulated structures each reflection class may be defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal cell.
Appears in list containing **_diffrn_refln_index_**. **Must** match parent data name **_diffrn_reflns_class_code**. **[diffrn_refln]**

_diffrn_refln_counts_bg_1
_diffrn_refln_counts_bg_2
_diffrn_refln_counts_net
_diffrn_refln_counts_peak
_diffrn_refln_counts_total (*numb*)
The diffractometer counts for the measurements: background before the peak, background after the peak, net counts after background removed, counts for peak scan or position, and the total counts (background plus peak).
Appears in list containing **_diffrn_refln_index_**. **[diffrn_refln]**

_diffrn_refln_crystal_id (*char*)
Code identifying each crystal if multiple crystals are used. Is used to link with **_exptl_crystal_id** in the **_exptl_crystal_** list.
Appears in list containing **_diffrn_refln_index_**. **Must** match parent data name **_exptl_crystal_id**. **[diffrn_refln]**

_diffrn_refln_detect_slit_horiz
_diffrn_refln_detect_slit_vert (*numb*)
Total slit apertures in degrees in the diffraction plane (*_**horiz**) and perpendicular to the diffraction plane (*_**vert**).
Appears in list containing **_diffrn_refln_index_**.
The permitted range is 0.0 → 90.0. **[diffrn_refln]**

_diffrn_refln_elapsed_time (*numb*)
Elapsed time in minutes from the start of the diffraction experiment to the measurement of this intensity.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is 0.0 → ∞. **[diffrn_refln]**

_diffrn_refln_index_h
_diffrn_refln_index_k
_diffrn_refln_index_l (*numb*)
Miller indices of a measured reflection. These need not match the **_refln_index_h**, ***_k**, ***_l** values if a transformation of the original measured cell has taken place. Details of the cell transformation are given in **_diffrn_reflns_reduction_process**. See also **_diffrn_reflns_transf_matrix_**.
Appears in list as essential element of loop structure. **[diffrn_refln]**

_diffrn_refln_intensity_net (*numb*)
Net intensity calculated from the diffraction counts after the attenuator and standard scales have been applied.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0 \rightarrow \infty$. **[diffrn_refln]**

_diffrn_refln_intensity_sigma (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_refln_intensity_u.
Standard uncertainty (e.s.d.) of the net intensity calculated from the diffraction counts after the attenuator and standard scales have been applied.
*Appears in list containing **_diffrn_refln_index_**.*
The permitted range is $0 \rightarrow \infty$. ***[diffrn_refln]***

_diffrn_refln_intensity_u (*numb*)
Standard uncertainty of the net intensity calculated from the diffraction counts after the attenuator and standard scales have been applied.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0 \rightarrow \infty$.
Related item: **_diffrn_refln_intensity_sigma** (alternate). **[diffrn_refln]**

_diffrn_refln_scale_group_code (*char*)
The code identifying the scale applicable to this reflection. This code must match with a specified **_diffrn_scale_group_code** value.
Appears in list containing **_diffrn_refln_index_**. **Must** match parent data name **_diffrn_scale_group_code**. **[diffrn_refln]**

_diffrn_refln_scan_mode (*char*)
The code identifying the mode of scanning for measurements using a diffractometer. See **_diffrn_refln_scan_width** and **_diffrn_refln_scan_mode_backgd.**
Appears in list containing **_diffrn_refln_index_**.
The data value must be one of the following:

om	ω scan
ot	$\omega/2\theta$ scan
q	Q scans (arbitrary reciprocal directions)

[diffrn_refln]

_diffrn_refln_scan_mode_backgd (*char*)
The code identifying the mode of scanning a reflection to measure the background intensity.
Appears in list containing **_diffrn_refln_index_**.
The data value must be one of the following:

st	stationary counter background
mo	moving counter background

[diffrn_refln]

_diffrn_refln_scan_rate (*numb*)
The rate of scanning a reflection in degrees per minute to measure the intensity.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_refln]**

_diffrn_refln_scan_time_backgd (*numb*)
The time spent measuring each background in seconds.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_refln]**

_diffrn_refln_scan_width (*numb*)
The scan width in degrees of the scan mode defined by the code **_diffrn_refln_scan_mode.**
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0.0 \rightarrow 90.0$. **[diffrn_refln]**

_diffrn_refln_sint/lambda (*numb*)
The $(\sin\theta)/\lambda$ value in reciprocal ångströms for this reflection.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_refln]**

_diffrn_refln_standard_code (*char*)
A code indicating that this reflection was measured as a standard reflection. The value must be '.' or match one of the **_diffrn_standard_refln_code** values.
Appears in list containing **_diffrn_refln_index_**. **Must** match parent data name **_diffrn_standard_refln_code**.
Examples: '1', '2', '3', 's1', 's2', 's3', 'A', 'B', 'C'. **[diffrn_refln]**

_diffrn_refln_wavelength (*numb*)
The mean wavelength in ångströms of the radiation used to measure the intensity of this reflection. This is an important parameter for reflections measured using energy-dispersive detectors or the Laue method.
Appears in list containing **_diffrn_refln_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_refln]**

_diffrn_refln_wavelength_id (*char*)
Code identifying the wavelength in the **_diffrn_radiation_** list.
Appears in list containing **_diffrn_refln_index_**. **Must** match parent data name **_diffrn_radiation_wavelength_id**.
Examples: 'x1', 'x2', 'neut'. **[diffrn_refln]**

DIFFRN_REFLNS

Data items in the DIFFRN_REFLNS category record details about the set of intensities measured in the diffraction experiment.

The DIFFRN_REFLNS data items specify the parameters that apply to all intensity measurements. The DIFFRN_REFLNS data items are not looped. (The DIFFRN_REFLN data items refer to individual intensity measurements and must be included in looped lists.)

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_diffrn_reflns_number              1592
_diffrn_reflns_av_R_equivalents    0
_diffrn_reflns_av_unetI/netI       .027
_diffrn_reflns_limit_h_min         0
_diffrn_reflns_limit_h_max         6
_diffrn_reflns_limit_k_min        -17
_diffrn_reflns_limit_k_max         0
_diffrn_reflns_limit_l_min         0
_diffrn_reflns_limit_l_max         22
_diffrn_reflns_theta_min           3.71
_diffrn_reflns_theta_max           61.97
```

_diffrn_reflns_av_R_equivalents (*numb*)
The residual $[\sum \mathrm{av}|\Delta(I)| / \sum |\mathrm{av}(I)|]$ for symmetry-equivalent reflections used to calculate the average intensity $\mathrm{av}(I)$. The $\mathrm{av}|\Delta(I)|$ term is the average absolute difference between $\mathrm{av}(I)$ and the individual symmetry-equivalent intensities.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_reflns]**

_diffrn_reflns_av_sigmaI/netI (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_reflns_av_unetI/netI.
Measure $[\sum |u(\mathrm{net}\, I)| / \sum |\mathrm{net}\, I|]$ *for all measured reflections.*
The permitted range is $0.0 \rightarrow \infty$. ***[diffrn_reflns]***

_diffrn_reflns_av_unetI/netI (*numb*)
Measure $[\sum |u(\mathrm{net}\, I)| / \sum |\mathrm{net}\, I|]$ for all measured reflections.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_diffrn_reflns_av_sigmaI/netI** (alternate). **[diffrn_reflns]**

_diffrn_reflns_limit_h_max
_diffrn_reflns_limit_h_min
_diffrn_reflns_limit_k_max
_diffrn_reflns_limit_k_min
_diffrn_reflns_limit_l_max
_diffrn_reflns_limit_l_min (*numb*)
The limits on the Miller indices of the intensities specified by **_diffrn_refln_index_h**, ***_k**, ***_l**.
[diffrn_reflns]

_diffrn_reflns_number (*numb*)
The total number of measured intensities, excluding reflections that are classed as systematically absent arising from translational symmetry in the crystal unit cell.
The permitted range is $0 \rightarrow \infty$. **[diffrn_reflns]**

_diffrn_reflns_reduction_process (*char*)
A description of the process used to reduce the intensities into structure-factor magnitudes.
Example: 'data averaged using Fisher test'. **[diffrn_reflns]**

_diffrn_reflns_resolution_full (*numb*)
The resolution in reciprocal ångströms at which the measured reflection count is close to complete.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_diffrn_reflns_theta_full** (alternate). **[diffrn_reflns]**

_diffrn_reflns_resolution_max (*numb*)
Maximum resolution in reciprocal ångströms of the measured diffraction pattern.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_diffrn_reflns_theta_max** (alternate). **[diffrn_reflns]**

_diffrn_reflns_theta_full (*numb*)
The θ angle (in degrees) at which the measured reflection count is close to complete. The fraction of unique reflections measured out to this angle is given by **_diffrn_measured_fraction_theta_full**.
The permitted range is $0.0 \rightarrow 90.0$. **[diffrn_reflns]**

_diffrn_reflns_theta_max (*numb*)
Maximum θ angle in degrees for the measured intensities.
The permitted range is $0.0 \rightarrow 90.0$. **[diffrn_reflns]**

_diffrn_reflns_theta_min (*numb*)
Minimum θ angle in degrees for the measured intensities.
The permitted range is $0.0 \rightarrow 90.0$. **[diffrn_reflns]**

_diffrn_reflns_transf_matrix_11
_diffrn_reflns_transf_matrix_12
_diffrn_reflns_transf_matrix_13
_diffrn_reflns_transf_matrix_21
_diffrn_reflns_transf_matrix_22
_diffrn_reflns_transf_matrix_23
_diffrn_reflns_transf_matrix_31
_diffrn_reflns_transf_matrix_32
_diffrn_reflns_transf_matrix_33 (*numb*)
Elements of the matrix used to transform the diffraction reflection indices **_diffrn_refln_index_h**, ***_k**, ***_l** into the **_refln_index_h**, ***_k**, ***_l** indices.

$$(\,h \quad k \quad l\,)_{\mathrm{diffraction}} \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} = (\,h' \quad k' \quad l'\,)\,.$$

[diffrn_reflns]

DIFFRN_REFLNS_CLASS

Data items in the DIFFRN_REFLNS_CLASS category record details about the classes of reflections measured in the diffraction experiment.

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

Each reflection class is defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal lattice.

```
loop_
     _diffrn_reflns_class_number
     _diffrn_reflns_class_d_res_high
     _diffrn_reflns_class_d_res_low
     _diffrn_reflns_class_av_R_eq
     _diffrn_reflns_class_code
     _diffrn_reflns_class_description
 1580 0.551 6.136 0.015 'Main' 'm=0; main reflections'
 1045 0.551 6.136 0.010 'Sat1' 'm=1; first-order satellites'
```

_diffrn_reflns_class_av_R_eq (*numb*)
For each reflection class, the residual $[\sum \mathrm{av}|\Delta(I)| / \sum |\mathrm{av}(I)|]$ for symmetry-equivalent reflections used to calculate the average intensity $\mathrm{av}(I)$. The $\mathrm{av}|\Delta(I)|$ term is the average absolute difference between $\mathrm{av}(I)$ and the individual symmetry-equivalent intensities.
Appears in list containing **_diffrn_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_reflns_class]**

_diffrn_reflns_class_av_sgI/I (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_reflns_class_av_uI/I.
Measure $[\sum |u(\mathrm{net}\, I)| / \sum |\mathrm{net}\, I|]$ *for all measured intensities in a reflection class.*
Appears in list containing ***_diffrn_reflns_class_code****.*
The permitted range is $0.0 \rightarrow \infty$. ***[diffrn_reflns_class]***

_diffrn_reflns_class_av_uI/I (*numb*)
Measure $[\sum |u(\mathrm{net}I)| / \sum |\mathrm{net}I|]$ for all measured intensities in a reflection class.
Appears in list containing **_diffrn_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_diffrn_reflns_class_av_sgI/I** (alternate).
[diffrn_reflns_class]

_diffrn_reflns_class_code (*char*)
The code identifying a certain reflection class.
Appears in list as essential element of loop structure. May match child data name(s): **_diffrn_refln_class_code**.
Examples: '1', 'm1', 's2'. **[diffrn_reflns_class]**

_diffrn_reflns_class_d_res_high (*numb*)
The smallest value in ångströms of the interplanar spacings of the reflections in each reflection class. This is called the highest resolution for this reflection class.
Appears in list containing **_diffrn_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_reflns_class]**

_diffrn_reflns_class_d_res_low (*numb*)
The highest value in ångströms of the interplanar spacings of the reflections in each reflection class. This is called the lowest resolution for this reflection class.
Appears in list containing **_diffrn_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_reflns_class]**

_diffrn_reflns_class_description (*char*)
Description of each reflection class.
Appears in list containing **_diffrn_reflns_class_code**.
Examples: 'm=1 first order satellites', 'H0L0 common projection reflections'. **[diffrn_reflns_class]**

_diffrn_reflns_class_number (*numb*)
The total number of measured intensities for each reflection class, excluding the systematic absences arising from centring translations.
Appears in list containing **_diffrn_reflns_class_code**.
The permitted range is $0 \rightarrow \infty$. **[diffrn_reflns_class]**

DIFFRN_SCALE_GROUP

Data items in the DIFFRN_SCALE_GROUP category record details of the scaling factors applied to place all intensities in the reflection lists on a common scale. Scaling groups might, for instance, correspond to each film in a multi-film data set or each crystal in a multi-crystal data set.

Example 1 – hypothetical example.

```
loop_
_diffrn_scale_group_code
_diffrn_scale_group_I_net
  1    .86473
  2   1.0654
```

_diffrn_scale_group_code (*char*)
The code identifying a specific measurement group (*e.g.* for multi-film or multi-crystal data). The code must match a **_diffrn_refln_scale_group_code** in the reflection list.
Appears in list as essential element of loop structure. May match child data name(s): **_diffrn_refln_scale_group_code**.
Examples: '1', '2', '3', 's1', 'A', 'B', 'c1', 'c2', 'c3'. **[diffrn_scale_group]**

_diffrn_scale_group_I_net (*numb*)
The scale for a specific measurement group which is to be multiplied with the net intensity to place all intensities in the **_diffrn_refln_** or **_refln_** list on a common scale.
Appears in list containing **_diffrn_scale_group_code**.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_scale_group]**

DIFFRN_SOURCE

Data items in the DIFFRN_SOURCE category record details of the source of radiation used in the diffraction experiment.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_source              'rotating anode X-ray tube'
_diffrn_source_type         'Rigaku RU-200'
_diffrn_source_power        50
_diffrn_source_current      180
_diffrn_source_size         '8 mm x 0.4 mm broad focus'
```

_diffrn_radiation_source (*char*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_source.
The source of radiation.
[diffrn_source]

_diffrn_source (*char*)
The general class of the source of radiation.
Related item: **_diffrn_radiation_source** (alternate).
Examples: 'sealed X-ray tube', 'nuclear reactor', 'spallation source', 'electron microscope', 'rotating-anode X-ray tube', 'synchrotron'.
[diffrn_source]

_diffrn_source_current (*numb*)
The current in milliamperes at which the radiation source was operated.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_source]**

_diffrn_source_details (*char*)
A description of special aspects of the source.
[diffrn_source]

_diffrn_source_power (*numb*)
The power in kilowatts at which the radiation source was operated.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_source]**

_diffrn_source_size (*char*)
The dimensions of the source as viewed from the sample.
Examples: '8mm x 0.4 mm fine-focus', 'broad focus'. **[diffrn_source]**

_diffrn_source_take-off_angle (*numb*)
The complement of the angle in degrees between the normal to the surface of the X-ray tube target and the primary X-ray beam for beams generated by traditional X-ray tubes.
The permitted range is $0 \rightarrow 90$.
Example: '1.53'. **[diffrn_source]**

_diffrn_source_target (*char*)
The chemical element symbol for the X-ray target (usually the anode) used to generate X-rays. This can also be used for spallation sources.
The data value must be one of the following:

```
H  He Li Be B  C  N  O  F  Ne Na Mg Al Si P
S  Cl Ar K  Ca Sc Ti V  Cr Mn Fe Co Ni Cu Zn
Ga Ge As Se Br Kr Rb Sr Y  Zr Nb Mo Tc Ru Rh
Pd Ag Cd In Sn Sb Te I  Xe Cs Ba La Ce Pr Nd
Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu Hf Ta W  Re
Os Ir Pt Au Hg Tl Pb Bi Po At Rn Fr Ra Ac Th
Pa U  Np Pu Am Cm Bk Cf Es Fm Md No Lr
```

[diffrn_source]

_diffrn_source_type (*char*)
The make, model or name of the source of radiation.
Examples: 'NSLS beamline X8C', 'Rigaku RU200'. **[diffrn_source]**

_diffrn_source_voltage (*numb*)
The voltage in kilovolts at which the radiation source was operated.
The permitted range is $0.0 \rightarrow \infty$. **[diffrn_source]**

DIFFRN_STANDARD_REFLN

Data items in the DIFFRN_STANDARD_REFLN category record details about the reflections treated as standards during the measurement of the diffraction intensities. Note that these are the individual standard reflections, not the results of the analysis of the standard reflections.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_diffrn_standard_refln_index_h
_diffrn_standard_refln_index_k
_diffrn_standard_refln_index_l
3 2 4    1 9 1    3 0 10
```

_diffrn_standard_refln_code (*char*)
The code identifying a reflection measured as a standard reflection with the indices **_diffrn_standard_refln_index_**. This is the same code as the **_diffrn_refln_standard_code** in the **_diffrn_refln_** list.
Appears in list containing **_diffrn_standard_refln_index_**. May match child data name(s): **_diffrn_refln_standard_code**.
Examples: '1', '2', '3', 's1', 'A', 'B'. **[diffrn_standard_refln]**

_diffrn_standard_refln_index_h
_diffrn_standard_refln_index_k
_diffrn_standard_refln_index_l (*numb*)
Miller indices of standard reflections used in the diffraction measurement process.
Appears in list as essential element of loop structure. **[diffrn_standard_refln]**

DIFFRN_STANDARDS

Data items in the DIFFRN_STANDARDS category record details about the set of standard reflections used to monitor intensity stability during the measurement of diffraction intensities. Note that these records describe properties common to the set of standard reflections, not the standard reflections themselves.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_diffrn_standards_number           3
_diffrn_standards_interval_time    120
_diffrn_standards_decay_%          0
```

_diffrn_standards_decay_% (*numb, su*)
The percentage decrease in the mean of the intensities for the set of standard reflections from the start of the measurement process to the end. This value usually affords a measure of the overall decay in crystal quality during the diffraction measurement process. Negative values are used in exceptional instances where the final intensities are greater than the initial ones.
The permitted range is $-\infty \rightarrow 100$. **[diffrn_standards]**

_diffrn_standards_interval_count
_diffrn_standards_interval_time (*numb*)
The number of reflection intensities, or the time in minutes, between the measurement of standard reflection intensities.
The permitted range is $0 \rightarrow \infty$. **[diffrn_standards]**

_diffrn_standards_number (*numb*)
The number of unique standard reflections used during the measurement of the diffraction intensities.
The permitted range is $0 \rightarrow \infty$. **[diffrn_standards]**

_diffrn_standards_scale_sigma (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _diffrn_standards_scale_u.
The standard uncertainty (e.s.d.) of the individual mean standard scales applied to the intensity data.
The permitted range is $0.0 \rightarrow \infty$. ***[diffrn_standards]***

_diffrn_standards_scale_u (*numb*)
The standard uncertainty of the individual mean standard scales applied to the intensity data.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_diffrn_standards_scale_sigma** (alternate).
[diffrn_standards]

EXPTL

Data items in the EXPTL category record details about the experimental work prior to the intensity measurements and details about the absorption-correction technique employed.

*Example 1 – based on a paper by Steiner [Acta Cryst. (1996), C**52**, 2554–2556].*

```
_exptl_absorpt_coefficient_mu    0.962
_exptl_absorpt_correction_type   psi-scan
_exptl_absorpt_process_details
          'North, Phillips & Mathews (1968)'
_exptl_absorpt_correction_T_min  0.929
_exptl_absorpt_correction_T_max  0.997
```

_exptl_absorpt_coefficient_mu (*numb*)
The absorption coefficient μ in reciprocal millimetres calculated from the atomic content of the cell, the density and the radiation wavelength.
The permitted range is $0.0 \rightarrow \infty$. **[exptl]**

_exptl_absorpt_correction_T_max
_exptl_absorpt_correction_T_min (*numb*)
The maximum and minimum transmission factors for the crystal and radiation. These factors are also referred to as the absorption correction A or $1/A^*$.
The permitted range is $0.0 \rightarrow 1.0$. **[exptl]**

_exptl_absorpt_correction_type (*char*)
The absorption-correction type and method. The value 'empirical' should *not* be used unless more detailed information is not available.
The data value must be one of the following:

analytical	analytical from crystal shape
cylinder	cylindrical
empirical	empirical from intensities
gaussian	Gaussian from crystal shape
integration	integration from crystal shape
multi-scan	symmetry-related measurements
none	no absorption correction applied
numerical	numerical from crystal shape
psi-scan	ψ-scan corrections
refdelf	refined from ΔF
sphere	spherical

[exptl]

_exptl_absorpt_process_details (*char*)
Description of the absorption process applied to the intensities. A literature reference should be supplied for ψ-scan techniques.
Examples: 'Tompa analytical', 'MolEN (Fair, 1990)', '(North, Phillips & Mathews, 1968)'. **[exptl]**

_exptl_crystals_number (*numb*)
The total number of crystals used for the measurement of intensities.
The permitted range is $1 \rightarrow \infty$. **[exptl]**

_exptl_special_details (*char*)
Any special information about the experimental work prior to the intensity measurements. See also **_exptl_crystal_preparation**.
[exptl]

EXPTL_CRYSTAL

Data items in the EXPTL_CRYSTAL category record details about experimental measurements on the crystal or crystals used, such as shape, size or density.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_exptl_crystal_description          prism
_exptl_crystal_colour               colourless
_exptl_crystal_size_max             0.32
_exptl_crystal_size_mid             0.27
_exptl_crystal_size_min             0.10
_exptl_crystal_density_diffrn       1.146
_exptl_crystal_density_meas         ?
_exptl_crystal_density_method       'not measured'
_exptl_crystal_F_000                656
```

Example 2 – using separate items to define upper and lower limits for a value.

```
_exptl_crystal_density_meas_gt      2.5
_exptl_crystal_density_meas_lt      5.0
```

Example 3 – here the density was measured at some unspecified temperature below room temperature.

```
_exptl_crystal_density_meas_temp_lt 300
```

_exptl_crystal_colour (*char*)
The colour of the crystal.
May appear in list containing **_exptl_crystal_id**.
Related items:
_exptl_crystal_colour_lustre (alternate),
_exptl_crystal_colour_modifier (alternate),
_exptl_crystal_colour_primary (alternate).
Example: 'dark green'. **[exptl_crystal]**

_exptl_crystal_colour_lustre (*char*)
The enumeration list of standardized names developed for the International Centre for Diffraction Data. The colour of a crystal is given by the combination of **_exptl_crystal_colour_modifier** with **_exptl_crystal_colour_primary**, as in 'dark-green' or 'bluish-violet', if necessary combined with **_exptl_crystal_colour_lustre**, as in 'metallic-green'.
May appear in list containing **_exptl_crystal_id**.
Related item: **_exptl_crystal_colour** (alternate).
The data value must be one of the following:

metallic
dull
clear

[exptl_crystal]

_exptl_crystal_colour_modifier (*char*)
The enumeration list of standardized names developed for the International Centre for Diffraction Data. The colour of a crystal is given by the combination of **_exptl_crystal_colour_modifier** with **_exptl_crystal_colour_primary**, as in 'dark-green'

or 'bluish-violet', if necessary combined with **_exptl_crystal_colour_lustre**, as in 'metallic-green'.

May appear in list containing **_exptl_crystal_id**.

Related item: **_exptl_crystal_colour** (alternate).

The data value must be one of the following:

```
light
dark
whitish
blackish
grayish
brownish
reddish
pinkish
orangish
yellowish
greenish
bluish
```

[exptl_crystal]

_exptl_crystal_colour_primary (*char*)

The enumeration list of standardized names developed for the International Centre for Diffraction Data. The colour of a crystal is given by the combination of **_exptl_crystal_colour_modifier** with **_exptl_crystal_colour_primary**, as in 'dark-green' or 'bluish-violet', if necessary combined with **_exptl_crystal_colour_lustre**, as in 'metallic-green'.

May appear in list containing **_exptl_crystal_id**.

Related item: **_exptl_crystal_colour** (alternate).

The data value must be one of the following:

```
colourless
white
black
gray
brown
red
pink
orange
yellow
green
blue
violet
```

[exptl_crystal]

_exptl_crystal_density_diffrn (*numb*)

Density values calculated from the crystal cell and contents. The units are megagrams per cubic metre (grams per cubic centimetre).

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$. **[exptl_crystal]**

_exptl_crystal_density_meas (*numb, su*)

Density values measured using standard chemical and physical methods. The units are megagrams per cubic metre (grams per cubic centimetre).

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$. **[exptl_crystal]**

_exptl_crystal_density_meas_gt (*numb*)

The value above which the density measured using standard chemical and physical methods lies. The units are megagrams per cubic metre (grams per cubic centimetre). **_exptl_crystal_density_meas_gt** and **_exptl_crystal_density_meas_lt** should not be used to report new experimental work, for which **_exptl_crystal_density_meas** should be used. These items are intended for use in reporting information in existing databases and archives which would be misleading if reported under **_exptl_crystal_density_meas.**

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_exptl_crystal_density_meas** (alternate).

Example: '2.5' (lower limit for the density (only the range within which the density lies was given in the original paper)). **[exptl_crystal]**

_exptl_crystal_density_meas_lt (*numb*)

The value below which the density measured using standard chemical and physical methods lies. The units are megagrams per cubic metre (grams per cubic centimetre). **_exptl_crystal_density_meas_gt** and **_exptl_crystal_density_meas_lt** should not be used to report new experimental work, for which **_exptl_crystal_density_meas** should be used. These items are intended for use in reporting information in existing databases and archives which would be misleading if reported under **_exptl_crystal_density_meas.**

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_exptl_crystal_density_meas** (alternate).

Examples: '1.0' (specimen floats in water), '5.0' (upper limit for the density (only the range within which the density lies was given in the original paper)). **[exptl_crystal]**

_exptl_crystal_density_meas_temp (*numb, su*)

Temperature in kelvins at which **_exptl_crystal_density_meas** was determined.

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$. **[exptl_crystal]**

_exptl_crystal_density_meas_temp_gt (*numb*)

Temperature in kelvins above which **_exptl_crystal_density_meas** was determined. **_exptl_crystal_density_meas_temp_gt** and **_exptl_crystal_density_meas_temp_lt** should not be used for reporting new work, for which the correct temperature of measurement should be given. These items are intended for use in reporting information stored in databases or archives which would be misleading if reported under **_exptl_crystal_density_meas_temp.**

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_exptl_crystal_density_meas_temp** (alternate).

[exptl_crystal]

_exptl_crystal_density_meas_temp_lt (*numb*)

Temperature in kelvins below which **_exptl_crystal_density_meas** was determined. **_exptl_crystal_density_meas_temp_gt** and **_exptl_crystal_density_meas_temp_lt** should not be used for reporting new work, for which the correct temperature of measurement should be given. These items are intended for use in reporting information stored in databases or archives which would be misleading if reported under **_exptl_crystal_density_meas_temp.**

May appear in list containing **_exptl_crystal_id**.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_exptl_crystal_density_meas_temp** (alternate).

Example: '300' (The density was measured at some unspecified temperature below room temperature.). **[exptl_crystal]**

_exptl_crystal_density_method (*char*)
The method used to measure **_exptl_crystal_density_meas**.
May appear in list containing **_exptl_crystal_id**.
Examples: 'flotation in aqueous KI', 'not measured', 'Berman density torsion balance'. **[exptl_crystal]**

_exptl_crystal_description (*char*)
A description of the quality and habit of the crystal. The crystal dimensions should not normally be reported here; use instead **_exptl_crystal_size_** for the gross dimensions of the crystal and **_exptl_crystal_face_** to describe the relationship between individual faces.
May appear in list containing **_exptl_crystal_id**. **[exptl_crystal]**

_exptl_crystal_F_000 (*numb*)
The effective number of electrons in the crystal unit cell contributing to $F(000)$. This may contain dispersion contributions and is calculated as

$$F(000) = \left[\left(\sum f_r\right)^2 + \left(\sum f_i\right)^2\right]^{1/2},$$

where f_r = real part of the scattering factors at $\theta = 0°$, f_i = imaginary part of the scattering factors at $\theta = 0°$ and the sum is taken over each atom in the unit cell.
May appear in list containing **_exptl_crystal_id**.
The permitted range is $0.0 \rightarrow \infty$. **[exptl_crystal]**

_exptl_crystal_id (*char*)
Code identifying each crystal if multiple crystals are used. It is used to link with **_diffrn_refln_crystal_id** in the intensity measurements and with **_refln_crystal_id** in the **_refln_** list.
Appears in list as essential element of loop structure. May match child data name(s): **_diffrn_refln_crystal_id**, **_refln_crystal_id**. **[exptl_crystal]**

_exptl_crystal_preparation (*char*)
Details of crystal growth and preparation of the crystal (*e.g.* mounting) prior to the intensity measurements.
May appear in list containing **_exptl_crystal_id**.
Example: 'mounted in an argon-filled quartz capillary'. **[exptl_crystal]**

_exptl_crystal_pressure_history (*char*)
Relevant details concerning the pressure history of the sample.
May appear in list containing **_exptl_crystal_id**. **[exptl_crystal]**

_exptl_crystal_recrystallization_method (*char*)
Describes the method used to recrystallize the sample. Sufficient details should be given for the procedure to be repeated. The temperature or temperatures should be given as well as details of the solvent, flux or carrier gas with concentrations or pressures and ambient atmosphere. **[exptl_crystal]**

_exptl_crystal_size_length
_exptl_crystal_size_max
_exptl_crystal_size_mid
_exptl_crystal_size_min
_exptl_crystal_size_rad (*numb*)
The maximum, medial and minimum dimensions in millimetres of the crystal. If the crystal is a sphere, then the ***_rad** item is its radius. If the crystal is a cylinder, then the ***_rad** item is its radius and the ***_length** item is its length. These may appear in a list with **_exptl_crystal_id** if multiple crystals are used in the experiment.
May appear in list containing **_exptl_crystal_id**.
The permitted range is $0.0 \rightarrow \infty$. **[exptl_crystal]**

_exptl_crystal_thermal_history (*char*)
Relevant details concerning the thermal history of the sample.
May appear in list containing **_exptl_crystal_id**. **[exptl_crystal]**

EXPTL_CRYSTAL_FACE

Data items in the EXPTL_CRYSTAL_FACE category record details of the crystal faces.

Example 1 – based on structure PAWD2 of Vittal & Dean [Acta Cryst. (1996), C52, 1180–1182].

```
loop_
_exptl_crystal_face_index_h
_exptl_crystal_face_index_k
_exptl_crystal_face_index_l
_exptl_crystal_face_perp_dist
     0   -1   -2    .18274
     1    0   -2    .17571
    -1    1   -2    .17845
    -2    1    0    .21010
    -1    0    2    .18849
     1   -1    2    .20605
     2   -1    0    .24680
    -1    2    0    .19688
     0    1    2    .15206
```

_exptl_crystal_face_diffr_chi
_exptl_crystal_face_diffr_kappa
_exptl_crystal_face_diffr_phi
_exptl_crystal_face_diffr_psi (*numb*)
The goniometer angle settings in degrees when the perpendicular to the specified crystal face is aligned along a specified direction (*e.g.* the bisector of the incident and reflected beams in an optical goniometer).
Appears in list containing **_exptl_crystal_face_index_**. **[exptl_crystal_face]**

_exptl_crystal_face_index_h
_exptl_crystal_face_index_k
_exptl_crystal_face_index_l (*numb*)
Miller indices of the crystal face associated with the value **_exptl_crystal_face_perp_dist**.
Appears in list as essential element of loop structure. **[exptl_crystal_face]**

_exptl_crystal_face_perp_dist (*numb*)
The perpendicular distance in millimetres from the face to the centre of rotation of the crystal.
Appears in list containing **_exptl_crystal_face_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[exptl_crystal_face]**

GEOM

Data items in the GEOM and related (GEOM_ANGLE, GEOM_BOND, GEOM_CONTACT, GEOM_HBOND and GEOM_TORSION) categories record details about the molecular and crystal geometry as calculated from the ATOM, CELL and SYMMETRY data.

Geometry data are usually redundant, in that they can be calculated from other more fundamental quantities in the data block. However, they serve the dual purposes of providing a check on the correctness of both sets of data and of enabling the most important geometric data to be identified for publication by setting the appropriate publication flag.

*Example 1 – based on data set bagan of Yamane & DiSalvo [Acta Cryst. (1996), C**52**, 760–761].*

```
     _geom_special_details
;  All esds (except the esd in the dihedral angle between
   two l.s. planes) are estimated using the full covariance
   matrix. The cell esds are taken into account individually
   in the estimation of esds in distances, angles and
   torsion angles; correlations between esds in cell
   parameters are only used when they are defined by crystal
   symmetry. An approximate (isotropic) treatment of cell
   esds is used for estimating esds involving l.s. planes.
;
```

_geom_special_details (*char*)
The description of geometrical information not covered by the existing data names in the geometry categories, such as least-squares planes.

[geom]

GEOM_ANGLE

Data items in the GEOM_ANGLE category record details about the bond angles as calculated from the ATOM, CELL and SYMMETRY data.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_geom_angle_atom_site_label_1
_geom_angle_atom_site_label_2
_geom_angle_atom_site_label_3
_geom_angle
_geom_angle_site_symmetry_1
_geom_angle_site_symmetry_2
_geom_angle_site_symmetry_3
_geom_angle_publ_flag
 C2  O1  C5   111.6(2)  1_555  1_555  1_555  yes
 O1  C2  C3   110.9(2)  1_555  1_555  1_555  yes
 O1  C2  O21  122.2(3)  1_555  1_555  1_555  yes
 C3  C2  O21  127.0(3)  1_555  1_555  1_555  yes
 C2  C3  N4   101.3(2)  1_555  1_555  1_555  yes
 C2  C3  C31  111.3(2)  1_555  1_555  1_555  yes
 C2  C3  H3   107(1)    1_555  1_555  1_555  no
 N4  C3  C31  116.7(2)  1_555  1_555  1_555  yes
# - - - - data truncated for brevity - - - -
```

_geom_angle (*numb, su*)
Angle in degrees defined by the three sites **_geom_angle_atom_site_label_1**, *_2 and *_3. The site at *_2 is at the apex of the angle.

Appears in list containing **_geom_angle_atom_site_label_**. **[geom_angle]**

_geom_angle_atom_site_label_1
_geom_angle_atom_site_label_2
_geom_angle_atom_site_label_3 (*char*)
The labels of the three atom sites which define the angle given by **_geom_angle**. These must match labels specified as **_atom_site_label** in the atom list. Label 2 identifies the site at the apex of the angle.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[geom_angle]**

_geom_angle_publ_flag (*char*)
This code signals whether the angle is referred to in a publication or should be placed in a table of significant angles.

Appears in list containing **_geom_angle_atom_site_label_**.
The data value must be one of the following:

no	do not include angle in special list
n	abbreviation for 'no'
yes	do include angle in special list
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[geom_angle]**

_geom_angle_site_symmetry_1
_geom_angle_site_symmetry_2
_geom_angle_site_symmetry_3 (*char*)
The symmetry code of each atom site as the symmetry-equivalent position number '*n*' and the cell translation number '*klm*'. These numbers are combined to form the code '*n klm*' or *n_klm*. The character string *n_klm* is composed as follows: *n* refers to the symmetry operation that is applied to the coordinates stored in **_atom_site_fract_x**, **_atom_site_fract_y** and **_atom_site_fract_z**. It must match a number given in **_space_group_symop_id**. k, l and m refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the angle. These translations (x, y, z) are related to (k, l, m) by the relations $k = 5 + x$, $l = 5 + y$, $m = 5 + z$. By adding 5 to the translations, the use of negative numbers is avoided.

Appears in list containing **_geom_angle_atom_site_label_**.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_angle]**

GEOM_BOND

Data items in the GEOM_BOND category record details about bonds as calculated from the ATOM, CELL and SYMMETRY data.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_geom_bond_atom_site_label_1
_geom_bond_atom_site_label_2
_geom_bond_distance
_geom_bond_site_symmetry_1
_geom_bond_site_symmetry_2
_geom_bond_publ_flag
  O1  C2   1.342(4)  1_555  1_555  yes
  O1  C5   1.439(3)  1_555  1_555  yes
  C2  C3   1.512(4)  1_555  1_555  yes
  C2  O21  1.199(4)  1_555  1_555  yes
  C3  N4   1.465(3)  1_555  1_555  yes
  C3  C31  1.537(4)  1_555  1_555  yes
  C3  H3   1.00(3)   1_555  1_555  no
  N4  C5   1.472(3)  1_555  1_555  yes
# - - - - data truncated for brevity - - - -
```

_geom_bond_atom_site_label_1
_geom_bond_atom_site_label_2 (*char*)
The labels of two atom sites that form a bond. These must match labels specified as **_atom_site_label** in the atom list.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[geom_bond]**

_geom_bond_distance (*numb, su*)
The intramolecular bond distance in ångströms.

Appears in list containing **_geom_bond_atom_site_label_**.
The permitted range is $0.0 \rightarrow \infty$. **[geom_bond]**

_geom_bond_publ_flag (*char*)

This code signals whether the bond distance is referred to in a publication or should be placed in a list of significant bond distances.

Appears in list containing **_geom_bond_atom_site_label_**.

The data value must be one of the following:

no	do not include bond in special list
n	abbreviation for 'no'
yes	do include bond in special list
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[geom_bond]**

_geom_bond_site_symmetry_1
_geom_bond_site_symmetry_2 (*char*)

The symmetry code of each atom site as the symmetry-equivalent position number '*n*' and the cell translation number '*klm*'. These numbers are combined to form the code '*n klm*' or *n_klm*. The character string *n_klm* is composed as follows: *n* refers to the symmetry operation that is applied to the coordinates stored in **_atom_site_fract_x**, **_atom_site_fract_y** and **_atom_site_fract_z**. It must match a number given in **_space_group_symop_id**. *k*, *l* and *m* refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the bond. These translations (x, y, z) are related to (k, l, m) by the relations $k = 5 + x$, $l = 5 + y$, $m = 5 + z$. By adding 5 to the translations, the use of negative numbers is avoided.

Appears in list containing **_geom_bond_atom_site_label_**.

Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_bond]**

_geom_bond_valence (*numb*)

The bond valence calculated from **_geom_bond_distance**.

Appears in list containing **_geom_bond_atom_site_label_**. **[geom_bond]**

GEOM_CONTACT

Data items in the GEOM_CONTACT category record details about interatomic contacts as calculated from the ATOM, CELL and SYMMETRY data.

*Example 1 – based on data set CLPHO6 of Ferguson, Ruhl, McKervey & Browne [Acta Cryst. (1992), C**48**, 2262–2264].*

```
loop_
 _geom_contact_atom_site_label_1
 _geom_contact_atom_site_label_2
 _geom_contact_distance
 _geom_contact_site_symmetry_1
 _geom_contact_site_symmetry_2
 _geom_contact_publ_flag
 O(1)  O(2)     2.735(3)  .  .  yes
 H(O1) O(2)     1.82      .  .  no
```

_geom_contact_atom_site_label_1
_geom_contact_atom_site_label_2 (*char*)

The labels of two atom sites that are within contact distance. The labels must match **_atom_site_label** codes in the atom list.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[geom_contact]**

_geom_contact_distance (*numb, su*)

The interatomic contact distance in ångströms.

Appears in list containing **_geom_contact_atom_site_label_**.

The permitted range is $0.0 \rightarrow \infty$. **[geom_contact]**

_geom_contact_publ_flag (*char*)

This code signals whether the contact distance is referred to in a publication or should be placed in a list of significant contact distances.

Appears in list containing **_geom_contact_atom_site_label_**.

The data value must be one of the following:

no	do not include distance in special list
n	abbreviation for 'no'
yes	do include distance in special list
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[geom_contact]**

_geom_contact_site_symmetry_1
_geom_contact_site_symmetry_2 (*char*)

The symmetry code of each atom site as the symmetry-equivalent position number '*n*' and the cell translation number '*klm*'. These numbers are combined to form the code '*n klm*' or *n_klm*. The character string *n_klm* is composed as follows: *n* refers to the symmetry operation that is applied to the coordinates stored in **_atom_site_fract_x**, **_atom_site_fract_y** and **_atom_site_fract_z**. It must match a number given in **_space_group_symop_id**. *k*, *l* and *m* refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the contact. These translations (x, y, z) are related to (k, l, m) by the relations $k = 5 + x$, $l = 5 + y$, $m = 5 + z$. By adding 5 to the translations, the use of negative numbers is avoided.

Appears in list containing **_geom_contact_atom_site_label_**.

Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_contact]**

GEOM_HBOND

Data items in the GEOM_HBOND category record details about hydrogen bonds as calculated from the ATOM, CELL and SYMMETRY data.

*Example 1 – based on $C_{14}H_{13}ClN_2O.H_2O$, reported by Palmer, Puddle & Lisgarten [Acta Cryst. (1993), C**49**, 1777–1779].*

```
loop_
_geom_hbond_atom_site_label_D
_geom_hbond_atom_site_label_H
_geom_hbond_atom_site_label_A
_geom_hbond_distance_DH
_geom_hbond_distance_HA
_geom_hbond_distance_DA
_geom_hbond_angle_DHA
_geom_hbond_publ_flag

N6   HN6  OW   0.888(8)  1.921(12)  2.801(8)  169.6(8)  yes
OW   HO2  O7   0.917(6)  1.923(12)  2.793(8)  153.5(8)  yes
OW   HO1  N10  0.894(8)  1.886(11)  2.842(8)  179.7(9)  yes
```

_geom_hbond_angle_DHA (*numb, su*)

Angle in degrees defined by the three sites **_geom_hbond_atom_site_label_D**, *_H and *_A. The site at *_H (the hydrogen atom participating in the interaction) is at the apex of the angle.

Appears in list containing **_geom_hbond_atom_site_label_**. **[geom_hbond]**

_geom_hbond_atom_site_label_D
_geom_hbond_atom_site_label_H
_geom_hbond_atom_site_label_A (*char*)

The labels of three atom sites (respectively, the donor atom, hydrogen atom and acceptor atom) participating in a hydrogen bond. These must match labels specified as **_atom_site_label** in the atom list.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[geom_hbond]**

_geom_hbond_distance_DH
_geom_hbond_distance_HA
_geom_hbond_distance_DA (*numb, su*)

Distances in ångströms between the donor and hydrogen (*_DH), hydrogen and acceptor (*_HA) and donor and acceptor (*_DA) sites in a hydrogen bond.

Appears in list containing **_geom_hbond_atom_site_label_**.

The permitted range is $0.0 \rightarrow \infty$. **[geom_hbond]**

_geom_hbond_publ_flag (*char*)

This code signals whether the hydrogen-bond information is referred to in a publication or should be placed in a table of significant hydrogen-bond geometry.

Appears in list containing **_geom_hbond_atom_site_label_**.

The data value must be one of the following:

no	do not include bond in special list
n	abbreviation for 'no'
yes	do include bond in special list
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[geom_hbond]**

_geom_hbond_site_symmetry_D
_geom_hbond_site_symmetry_H
_geom_hbond_site_symmetry_A (*char*)

The symmetry code of each atom site as the symmetry-equivalent position number '*n*' and the cell translation number '*klm*'. These numbers are combined to form the code '*n klm*' or *n_klm*. The character string *n_klm* is composed as follows: *n* refers to the symmetry operation that is applied to the coordinates stored in **_atom_site_fract_x**, **_atom_site_fract_y** and **_atom_site_fract_z**. It must match a number given in **_space_group_symop_id**. *k*, *l* and *m* refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the hydrogen bond. These translations (x, y, z) are related to (k, l, m) by the relations $k = 5 + x$, $l = 5 + y$, $m = 5 + z$. By adding 5 to the translations, the use of negative numbers is avoided.

Appears in list containing **_geom_hbond_atom_site_label_**.

Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_hbond]**

GEOM_TORSION

Data items in the GEOM_TORSION category record details about interatomic torsion angles as calculated from the ATOM, CELL and SYMMETRY data.

*Example 1 – based on data set CLPHO6 of Ferguson, Ruhl, McKervey & Browne [Acta Cryst. (1992), C**48**, 2262–2264].*

```
loop_
 _geom_torsion_atom_site_label_1
 _geom_torsion_atom_site_label_2
 _geom_torsion_atom_site_label_3
 _geom_torsion_atom_site_label_4
 _geom_torsion
 _geom_torsion_site_symmetry_1
 _geom_torsion_site_symmetry_2
 _geom_torsion_site_symmetry_3
 _geom_torsion_site_symmetry_4
 _geom_torsion_publ_flag
 C(9)  O(2)  C(7)   C(2)    71.8(2)  .  .  .  .      yes
 C(7)  O(2)  C(9)   C(10) -168.0(3)  .  .  .  2_666  yes
 C(10) O(3)  C(8)   C(6)  -167.7(3)  .  .  .  .      yes
 C(8)  O(3)  C(10)  C(9)   -69.7(2)  .  .  .  2_666  yes
 O(1)  C(1)  C(2)   C(3)  -179.5(4)  .  .  .  .      no
 O(1)  C(1)  C(2)   C(7)    -0.6(1)  .  .  .  .      no
```

_geom_torsion (*numb, su*)

The torsion angle in degrees bounded by the four atom sites identified by the **_geom_torsion_atom_site_label_** codes. These must match labels specified as **_atom_site_label** in the atom list. The torsion-angle definition should be that of Klyne and Prelog.

Reference: Klyne, W. & Prelog, V. (1960). *Experientia*, **16**, 521–523.

Appears in list containing **_geom_torsion_atom_site_label_**. **[geom_torsion]**

_geom_torsion_atom_site_label_1
_geom_torsion_atom_site_label_2
_geom_torsion_atom_site_label_3
_geom_torsion_atom_site_label_4 (*char*)

The labels of the four atom sites which define the torsion angle specified by **_geom_torsion**. These must match codes specified as **_atom_site_label** in the atom list. The torsion-angle definition should be that of Klyne and Prelog. The vector direction ***_label_2** to ***_label_3** is the viewing direction, and the torsion angle is the angle of twist required to superimpose the projection of the vector between site 2 and site 1 onto the projection of the vector between site 3 and site 4. Clockwise torsions are positive, anticlockwise torsions are negative.

Reference: Klyne, W. & Prelog, V. (1960). *Experientia*, **16**, 521–523.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[geom_torsion]**

_geom_torsion_publ_flag (*char*)

This code signals whether the torsion angle is referred to in a publication or should be placed in a table of significant torsion angles.

Appears in list containing **_geom_torsion_atom_site_label_**.

The data value must be one of the following:

no	do not include angle in special list
n	abbreviation for 'no'
yes	do include angle in special list
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[geom_torsion]**

_geom_torsion_site_symmetry_1
_geom_torsion_site_symmetry_2
_geom_torsion_site_symmetry_3
_geom_torsion_site_symmetry_4 (*char*)

The symmetry code of each atom site as the symmetry-equivalent position number '*n*' and the cell translation number '*klm*'. These numbers are combined to form the code '*n klm*' or *n_klm*. The character string *n_klm* is composed as follows: *n* refers to the symmetry operation that is applied to the coordinates stored in **_atom_site_fract_x**, **_atom_site_fract_y** and **_atom_site_fract_z**. It must match a number given in **_space_group_symop_id**. *k*, *l* and *m* refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the angle. These translations (x, y, z) are related to (k, l, m) by the relations $k = 5 + x$, $l = 5 + y$, $m = 5 + z$. By adding 5 to the translations, the use of negative numbers is avoided.

Appears in list containing **_geom_torsion_atom_site_label_**.

Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_torsion]**

JOURNAL

Data items in the JOURNAL category record details about the book-keeping by the journal staff when processing a CIF submitted for publication. The creator of a CIF will not normally specify these data items. The data names are not defined in the dictionary because they are for journal use only.

*Example 1 – based on Acta Cryst. file for entry HL0007 [Willis, Beckwith & Tozer (1991). Acta Cryst. C**47**, 2276–2277].*

```
_journal_date_recd_electronic    91-04-15
_journal_date_from_coeditor      91-04-18
_journal_date_accepted           91-04-18
_journal_date_printers_first     91-08-07
_journal_date_proofs_out         91-08-07
_journal_coeditor_code           HL0007
_journal_techeditor_code         C910963
_journal_coden_ASTM              ACSCEE
_journal_name_full       'Acta Crystallographica Section C'
_journal_year                    1991
_journal_volume                  47
_journal_issue                   NOV91
_journal_page_first              2276
_journal_page_last               2277
```

_journal_coden_ASTM
_journal_coden_Cambridge
_journal_coeditor_address
_journal_coeditor_code
_journal_coeditor_email
_journal_coeditor_fax
_journal_coeditor_name
_journal_coeditor_notes
_journal_coeditor_phone
_journal_data_validation_number
_journal_date_accepted
_journal_date_from_coeditor
_journal_date_to_coeditor
_journal_date_printers_final
_journal_date_printers_first
_journal_date_proofs_in
_journal_date_proofs_out
_journal_date_recd_copyright
_journal_date_recd_electronic
_journal_date_recd_hard_copy
_journal_issue
_journal_language
_journal_name_full
_journal_page_first
_journal_page_last
_journal_paper_category
_journal_suppl_publ_number
_journal_suppl_publ_pages
_journal_techeditor_address
_journal_techeditor_code
_journal_techeditor_email
_journal_techeditor_fax
_journal_techeditor_name
_journal_techeditor_notes
_journal_techeditor_phone
_journal_volume
_journal_year (*char*)

Data items specified by the journal staff.

[journal]

JOURNAL_INDEX

Data items in the JOURNAL_INDEX category are used to list terms used to generate the journal indexes. The creator of a CIF will not normally specify these data items.

*Example 1 – based on a paper by Zhu, Reynolds, Klein & Trudell [Acta Cryst. (1994), C**50**, 2067–2069].*

```
loop_
_journal_index_type
_journal_index_term
_journal_index_subterm
  O   C16H19NO4            .
  S   alkaloids           (-)-norcocaine
  S   (-)-norcocaine       .
  S
;  [2R,3S-(2\b,3\b)]-methyl
   3-(benzoyloxy)-8-azabicyclo[3.2.1]octane-2-carboxylate
;                          .
```

_journal_index_subterm
_journal_index_term
_journal_index_type (*char*)

Indexing terms supplied by the journal staff.

[journal_index]

PUBL

Data items in the PUBL category are used when submitting a manuscript for publication. They refer either to the paper as a whole, or to specific named elements within a paper (such as the title and abstract, or the *Comment* and *Experimental* sections of *Acta Crystallographica Section C*). The data items in the PUBL_BODY category should be used for the text of other submissions. Typically, each journal will supply a list of the specific items it requires in its *Notes for Authors*.

*Example 1 – based on Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_publ_section_title
;  trans-3-Benzoyl-2-(tert-butyl)-4-(iso-butyl)-
   1,3-oxazolidin-5-one
;

_publ_section_abstract
;  The oxazolidinone ring is a shallow envelope
   conformation with the tert-butyl and iso-butyl groups
   occupying trans-positions with respect to the ring. The
   angles at the N atom sum to 356.2\%, indicating a very
   small degree of pyramidalization at this atom. This is
   consistent with electron delocalization between the N
   atom and the carbonyl centre [N-C=O = 1.374(3)\%A].
;
```

*Example 2 – based on $C_{31}H_{48}N_4O_4$, reported by Coleman, Patrick, Andersen & Rettig [Acta Cryst. (1996), C**52**, 1525–1527].*

```
_publ_section_title
;  Hemiasterlin methyl ester
;

_publ_section_title_footnote
;  IUPAC name: methyl 2,5-dimethyl-4-2-[3-methyl-
   2-methylamino-3-(N-methylbenzo[b]pyrrol-
   3-yl)butanamido]-3,3-dimethyl-N-methyl-
   butanamido-2-hexenoate.
;
```

_publ_contact_author (*char*)

The name and address of the author submitting the manuscript and data block. This is the person contacted by the journal editorial staff. It is preferable to use the separate data items **_publ_contact_author_name** and **_publ_contact_author_address**.

Example:

```
; Professor George Ferguson
  Department of Chemistry and Biochemistry
  University of Guelph
  Ontario
  Canada
  N1G 2W1
;
```

[publ]

_publ_contact_author_address (*char*)
The address of the author submitting the manuscript and data block. This is the person contacted by the journal editorial staff.
Example:
```
; Department of Chemistry and Biochemistry
  University of Guelph
  Ontario
  Canada
  N1G 2W1
;
```
[publ]

_publ_contact_author_email (*char*)
E-mail address in a form recognizable to international networks. The format of e-mail addresses is given in Section 3.4, *Address Specification*, of *Internet Message Format*, RFC 2822, P. Resnick (Editor), Network Standards Group, April 2001.
Examples: '`name@host.domain.country`', '`bm@iucr.org`'. **[publ]**

_publ_contact_author_fax (*char*)
Facsimile telephone number of the author submitting the manuscript and data block. The recommended style is the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces. The earlier convention of including the international dialing prefix in parentheses is no longer recommended.
Examples: '`12(34)9477334`', '`12()349477334`'. **[publ]**

_publ_contact_author_id_iucr (*char*)
Identifier in the IUCr contact database of the author submitting the manuscript and data block. This identifier may be available from the *World Directory of Crystallographers* (http://wdc.iucr.org).
Example: '`2985`'. **[publ]**

_publ_contact_author_name (*char*)
The name of the author submitting the manuscript and data block. This is the person contacted by the journal editorial staff.
Example: '`Professor George Ferguson`'. **[publ]**

_publ_contact_author_phone (*char*)
Telephone number of the author submitting the manuscript and data block. The recommended style is the international dialing prefix, followed by the area code in parentheses, followed by the local number and any extension number prefixed by 'x', with no spaces. The earlier convention of including the international dialing prefix in parentheses is no longer recommended.
Examples: '`12(34)9477330`', '`12()349477330`', '`12(34)9477330x5543`'.
[publ]

_publ_contact_letter (*char*)
A letter submitted to the journal editor by the contact author.
[publ]

_publ_manuscript_creation (*char*)
A description of the word-processor package and computer used to create the word-processed manuscript stored as **_publ_manuscript_processed**.
Example: '`Tex file created by FrameMaker on a Sun 3/280`'. **[publ]**

_publ_manuscript_processed (*char*)
The full manuscript of a paper (excluding possibly the figures and the tables) output in ASCII characters from a word processor. Information about the generation of this data item must be specified in the data item **_publ_manuscript_creation**.
[publ]

_publ_manuscript_text (*char*)
The full manuscript of a paper (excluding figures and possibly the tables) output as standard ASCII text.
[publ]

_publ_requested_category (*char*)
The category of paper submitted. For submission to *Acta Crystallographica Section C* or *Acta Crystallographica Section E*, *only* the codes indicated for use with these journals should be used.
The data value must be one of the following:

`FA`	Full article
`FI`	Full submission – inorganic (*Acta C*)
`FO`	Full submission – organic (*Acta C*)
`FM`	Full submission – metal-organic (*Acta C*)
`CI`	CIF-access paper – inorganic (*Acta C*) (no longer in use)
`CO`	CIF-access paper – organic (*Acta C*) (no longer in use)
`CM`	CIF-access paper – metal-organic (*Acta C*) (no longer in use)
`EI`	Electronic submission – inorganic (*Acta E*)
`EO`	Electronic submission – organic (*Acta E*)
`EM`	Electronic submission – metal-organic (*Acta E*)
`AD`	Addenda and Errata (*Acta C*, *Acta E*)
`SC`	Short communication

Where no value is given, the assumed value is '`FA`'. **[publ]**

_publ_requested_coeditor_name (*char*)
The name of the co-editor whom the authors would like to handle the submitted manuscript.
[publ]

_publ_requested_journal (*char*)
The name of the journal to which the manuscript is being submitted.
[publ]

_publ_section_title
_publ_section_title_footnote
_publ_section_synopsis
_publ_section_abstract
_publ_section_comment
_publ_section_introduction
_publ_section_experimental
_publ_section_exptl_prep
_publ_section_exptl_refinement
_publ_section_exptl_solution
_publ_section_discussion
_publ_section_acknowledgements
_publ_section_references
_publ_section_figure_captions
_publ_section_table_legends (*char*)
The sections of a manuscript if submitted in parts. As an alternative, see **_publ_manuscript_text** and **_publ_manuscript_processed**. The **_publ_section_exptl_prep**, **_publ_section_exptl_refinement** and **_publ_section_exptl_solution** items are preferred for separating the chemical preparation, refinement and structure solution aspects of the experimental description.
[publ]

PUBL_AUTHOR

Data items in the PUBL_AUTHOR category record details of the authors of a manuscript submitted for publication.

*Example 1 – based on Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_publ_author_name
_publ_author_address

     'Willis, Anthony C.'
;    Research School of Chemistry
     Australian National University
     GPO Box 4
     Canberra, ACT
     Australia    2601
;
```

_publ_author_address (*char*)

The address of a publication author. If there is more than one author, this will be looped with **_publ_author_name**.

May appear in list containing **_publ_author_name**.

Example:

```
; Department
  Institute
  Street
  City and postcode
  COUNTRY
;
```

[publ_author]

_publ_author_email (*char*)

The e-mail address of a publication author. If there is more than one author, this will be looped with **_publ_author_name**. The format of e-mail addresses is given in Section 3.4, *Address Specification*, of *Internet Message Format*, RFC 2822, P. Resnick (Editor), Network Standards Group, April 2001.

May appear in list containing **_publ_author_name**.

Examples: 'name@host.domain.country', 'bm@iucr.org'. **[publ_author]**

_publ_author_footnote (*char*)

A footnote accompanying an author's name in the list of authors of a paper. Typically indicates sabbatical address, additional affiliations or date of decease.

May appear in list containing **_publ_author_name**.

Examples: 'On leave from U. Western Australia', 'Also at Department of Biophysics'. **[publ_author]**

_publ_author_id_iucr (*char*)

Identifier in the IUCr contact database of a publication author. This identifier may be available from the *World Directory of Crystallographers* (http://wdc.iucr.org).

May appear in list.

Example: '2985'. **[publ_author]**

_publ_author_name (*char*)

The name of a publication author. If there are multiple authors, this will be looped with **_publ_author_address**. The family name(s), followed by a comma and including any dynastic components, precedes the first names or initials.

May appear in list as essential element of loop structure.

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.', 'M\"uller, H.A.', 'Ross II, C.R.'.

[publ_author]

PUBL_BODY

Data items in the PUBL_BODY category permit the labelling of different text sections within the body of a paper. Note that these should not be used in a paper which has a standard format with sections tagged by specific data names (such as in *Acta Crystallographica Section C*). Typically, each journal will supply a list of the specific items it requires in its *Notes for Authors*.

*Example 1 – based on a paper by R. Restori & D. Schwarzenbach [Acta Cryst. (1996), A**52**, 369–378].*

```
loop_
_publ_body_element
_publ_body_label
_publ_body_title
_publ_body_format
_publ_body_contents

     section   1         Introduction                    cif
; X-ray diffraction from a crystalline material provides
  information on the thermally and spatially averaged
  electron density in the crystal...
;
     section   2         Theory                          tex
; In the rigid-atom approximation, the dynamic electron
  density of an atom is described by the convolution
  product of the static atomic density and a probability
  density function,
  $\rho_dyn(\bf r) = \rho_stat(\bf r) * P(\bf r). \eqno(1)$
;
```

*Example 2 – based on a paper by R. J. Papoular, Y. Vekhter & P. Coppens [Acta Cryst. (1996), A**52**, 397–407].*

```
loop_
_publ_body_element
_publ_body_label
_publ_body_title
_publ_body_contents

     section     3
; The two-channel method for retrieval of the deformation
  electron density
;
     .
     subsection  3.1  'The two-channel entropy S[\D\r(r)]'
; As the wide dynamic range involved in the total electron
  density...
;
     subsection  3.2
'Uniform vs informative prior model densities'        .
     subsubsection  3.2.1  'Use of uniform models'
; Straightforward algebra leads to expressions analogous
  to...
;
```

_publ_body_contents (*char*)

A text section of a paper.

Appears in list containing **_publ_body_label**. **[publ_body]**

_publ_body_element (*char*)

The functional role of the associated text section.

Appears in list containing **_publ_body_label**.

The data value must be one of the following:

- section
- subsection
- subsubsection
- appendix
- footnote

[publ_body]

_publ_body_format (*char*)

Code indicating the appropriate typesetting conventions for accented characters and special symbols in the text section.

Appears in list containing **_publ_body_label**.

The data value must be one of the following:

ascii	no coding for special symbols
cif	CIF convention
latex	LaTeX
rtf	Rich Text Format
sgml	SGML (ISO 8879)
tex	TeX
troff	troff or nroff

Where no value is given, the assumed value is 'cif'. **[publ_body]**

_publ_body_label (*char*)

Code identifying the section of text. The combination of this with **_publ_body_element** must be unique.

Appears in list as essential element of loop structure.

Examples: '1', '1.1', '2.1.3'. **[publ_body]**

_publ_body_title (*char*)

Title of the associated section of text.

Appears in list containing **_publ_body_label**. **[publ_body]**

PUBL_MANUSCRIPT_INCL

Data items in the PUBL_MANUSCRIPT_INCL category allow the authors of a manuscript submitted for publication to list data names that should be added to the standard request list used by the journal printing software. Although these fields are primarily intended to identify CIF data items that the author wishes to include in a published paper, they can also be used to identify data names created so that non-CIF items can be included in the publication. Note that *_item names *must* be enclosed in single quotes.

Example 1 – directive to include a hydrogen-bonding table, including cosmetic headings in comments.

```
loop_
_publ_manuscript_incl_extra_item
_publ_manuscript_incl_extra_info
_publ_manuscript_incl_extra_defn
#
# Include Hydrogen Bonding Geometry Description
# =============================================
# Name                              explanation    standard?
# ----                              -----------    ---------
  '_geom_hbond_atom_site_label_D'  'H-bond donor'     yes
  '_geom_hbond_atom_site_label_H'  'H-bond hydrogen'  yes
  '_geom_hbond_atom_site_label_A'  'H-bond acceptor'  yes
  '_geom_hbond_distance_DH'        'H-bond D-H'       yes
  '_geom_hbond_distance_HA'        'H-bond H...A'     yes
  '_geom_hbond_distance_DA'        'H-bond D...A'     yes
  '_geom_hbond_angle_DHA'          'H-bond D-H...A'   yes
```

Example 2 – hypothetical example including both standard CIF data items and a non-CIF quantity which the author wishes to list.

```
loop_
_publ_manuscript_incl_extra_item
_publ_manuscript_incl_extra_info
_publ_manuscript_incl_extra_defn
  '_atom_site_symmetry_multiplicity'
      'to emphasise special sites'       yes
  '_chemical_compound_source'
      'rare material, unusual source'    yes
  '_reflns_d_resolution_high'
      'limited data are a problem here'   yes
  '_crystal_magnetic_permeability'
      'unusual value for this material'  no
```

_publ_manuscript_incl_extra_defn (*char*)

Flags whether the corresponding data item marked for inclusion in a journal request list is a standard CIF definition or not.

Appears in list containing **_publ_manuscript_incl_extra_item**.

The data value must be one of the following:

no	not a standard CIF data name
n	abbreviation for 'no'
yes	a standard CIF data name
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'yes'. **[publ_manuscript_incl]**

_publ_manuscript_incl_extra_info (*char*)

A short note indicating the reason why the author wishes the corresponding data item marked for inclusion in the journal request list to be published.

Appears in list containing **_publ_manuscript_incl_extra_item**. **[publ_manuscript_incl]**

_publ_manuscript_incl_extra_item (*char*)

Specifies the inclusion of specific data into a manuscript which are not normally requested by the journal. The values of this item are the extra data names (which *must* be enclosed in single quotes) that will be added to the journal request list.

Appears in list as essential element of loop structure. **[publ_manuscript_incl]**

REFINE

Data items in the REFINE category record details about the structure-refinement parameters.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_refine_special_details    sfls:_F_calc_weight_full_matrix

_refine_ls_structure_factor_coef  F
_refine_ls_matrix_type            full
_refine_ls_weighting_scheme       calc
_refine_ls_weighting_details     'w=1/(u^2^(F)+0.0004F^2^)'
_refine_ls_hydrogen_treatment     refxyz
_refine_ls_extinction_method      Zachariasen
_refine_ls_extinction_coef        3514(42)
_refine_ls_extinction_expression
; Larson, A. C. (1970). "Crystallographic Computing", edited
  by F. R. Ahmed. Eq. (22) p. 292. Copenhagen: Munksgaard.
;
_refine_ls_abs_structure_details
; The absolute configuration was assigned to agree with that
  of its precursor l-leucine at the chiral centre C3.
;

_refine_ls_number_reflns          1408
_refine_ls_number_parameters      272
_refine_ls_number_restraints      0
_refine_ls_number_constraints     0
_refine_ls_R_factor_all           .038
_refine_ls_R_factor_gt            .034
_refine_ls_wR_factor_all          .044
_refine_ls_wR_factor_gt           .042
_refine_ls_goodness_of_fit_all   1.462
_refine_ls_goodness_of_fit_gt    1.515
_refine_ls_shift/su_max           .535
_refine_ls_shift/su_mean          .044
_refine_diff_density_min         -.108
_refine_diff_density_max          .131
```

_refine_diff_density_max
_refine_diff_density_min
_refine_diff_density_rms (*numb, su*)
The largest and smallest values and the root-mean-square deviation, in electrons per ångström cubed, of the final difference electron density. The *_**rms** value is measured with respect to the arithmetic mean density and is derived from summations over each grid point in the asymmetric unit of the cell. This quantity is useful for assessing the significance of *_**min** and *_**max** values, and also for defining suitable contour levels.

[refine]

_refine_ls_abs_structure_details (*char*)
The nature of the absolute structure and how it was determined.

[refine]

_refine_ls_abs_structure_Flack (*numb, su*)
The measure of absolute structure as defined by Flack (1983). For centrosymmetric structures, the only permitted value, if the data name is present, is 'inapplicable', represented by '.' . For noncentrosymmetric structures, the value must lie in the 99.97% Gaussian confidence interval $-3u \leq x \leq 1+3u$ and a standard uncertainty (e.s.d.) u must be supplied. The **_enumeration_range** of $0.0 \rightarrow 1.0$ is correctly interpreted as meaning $(0.0-3u) \leq x \leq (1.0+3u)$.

Reference: Flack, H. D. (1983). *Acta Cryst.* A**39**, 876–881.

The permitted range is $0.0 \rightarrow 1.0$. **[refine]**

_refine_ls_abs_structure_Rogers (*numb, su*)
The measure of absolute structure as defined by Rogers (1981). The value must lie in the 99.97% Gaussian confidence interval $-1-3u \leq \eta \leq 1+3u$ and a standard uncertainty (e.s.d.) u must be supplied. The **_enumeration_range** of $-1.0 \rightarrow 1.0$ is correctly interpreted as meaning $(-1.0-3u) \leq \eta \leq (1.0+3u)$.

Reference: Rogers, D. (1981). *Acta Cryst.* A**37**, 734–741.

The permitted range is $-1.0 \rightarrow 1.0$. **[refine]**

_refine_ls_d_res_high (*numb*)
The smallest value in ångströms of the interplanar spacings of the reflections used in the refinement. This is called the highest resolution.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_d_res_low (*numb*)
The largest value in ångströms of the interplanar spacings of the reflections used in the refinement. This is called the lowest resolution.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_extinction_coef (*numb, su*)
The extinction coefficient used to calculate the correction factor applied to the structure-factor data. The nature of the extinction coefficient is given in the definitions of **_refine_ls_extinction_expression** and **_refine_ls_extinction_method**. For the 'Zachariasen' method it is the r^* value; for the 'Becker–Coppens type 1 isotropic' method it is the 'g' value and for 'Becker–Coppens type 2 isotropic' corrections it is the 'ρ' value. Note that the magnitude of these values is usually of the order of 10 000.

References: Becker, P. J. & Coppens, P. (1974). *Acta Cryst.* A**30**, 129–147, 148–153. Zachariasen, W. H. (1967). *Acta Cryst.* **23**, 558–564. Larson, A. C. (1967). *Acta Cryst.* **23**, 664–665.

Example: '`3472(52)`' (Zachariasen coefficient r^* = 0.347(5) E04). **[refine]**

_refine_ls_extinction_expression (*char*)
A description of or reference to the extinction-correction equation used to apply the data item **_refine_ls_extinction_coef**. This information must be sufficient to reproduce the extinction-correction factors applied to the structure factors.

Example:

```
; Larson, A. C. (1970). "Crystallographic Computing",
  edited by F. R. Ahmed. Eq. (22), p. 292.
  Copenhagen: Munksgaard.
;
```
[refine]

_refine_ls_extinction_method (*char*)
A description of the extinction-correction method applied. This description should include information about the correction method, either 'Becker–Coppens' or 'Zachariasen'. The latter is sometimes referred to as the 'Larson' method even though it employs Zachariasen's formula. The Becker–Coppens procedure is referred to as 'type 1' when correcting secondary extinction dominated by the mosaic spread; as 'type 2' when secondary extinction is dominated by particle size and includes a primary extinction component; and as 'mixed' when there is a mixture of types 1 and 2. For the Becker–Coppens method, it is also necessary to set the mosaic distribution as either 'Gaussian' or 'Lorentzian' and the nature of the extinction as 'isotropic' or 'anisotropic'. Note that if either the 'mixed' or 'anisotropic' corrections are applied, the multiple coefficients cannot be contained in *_**extinction_coef** and must be listed in **_refine_special_details**.

References: Becker, P. J. & Coppens, P. (1974). *Acta Cryst.* A**30**, 129–147, 148–153. Zachariasen, W. H. (1967). *Acta Cryst.* **23**, 558–564. Larson, A. C. (1967). *Acta Cryst.* **23**, 664–665.

Where no value is given, the assumed value is '`Zachariasen`'.

Examples: '`B-C type 2 Gaussian isotropic`', '`none`'. **[refine]**

_refine_ls_goodness_of_fit_all (*numb, su*)
The least-squares goodness-of-fit parameter S for all reflections after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also **_refine_ls_restrained_S_** definitions.

$$S = \left(\frac{\sum |w| Y_{\text{obs}} - Y_{\text{calc}}|^2|}{N_{\text{ref}} - N_{\text{param}}} \right)^{1/2},$$

where Y_{obs} = the observed coefficients (see **_refine_ls_structure_factor_coef**), Y_{calc} = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight $(1/u^2)$, u = the standard uncertainty, N_{ref} = the number of reflections used in the refinement, N_{param} = the number of refined parameters and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_goodness_of_fit_gt (*numb, su*)

The least-squares goodness-of-fit parameter S for significantly intense reflections (see **_reflns_threshold_expression**) after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also **_refine_ls_restrained_S_** definitions.

$$S = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{N_{\rm ref} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine_ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight $(1/u^2)$, u = the standard uncertainty, $N_{\rm ref}$ = the number of reflections used in the refinement, $N_{\rm param}$ = the number of refined parameters; the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_goodness_of_fit_obs** (alternate). **[refine]**

_refine_ls_goodness_of_fit_obs (*numb, su*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refine_ls_goodness_of_fit_gt.

*The least-squares goodness-of-fit parameter S for observed reflections (see **_reflns_observed_criterion**) after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also **_refine_ls_restrained_S_** definitions.*

$$S = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{N_{\rm ref} - N_{\rm param}} \right)^{1/2},$$

*where $Y_{\rm obs}$ = the observed coefficients (see **_refine_ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight $(1/u^2)$, u = the standard uncertainty, $N_{\rm ref}$ = the number of reflections used in the refinement, $N_{\rm param}$ = the number of refined parameters; the sum is taken over the specified reflections.*

The permitted range is $0.0 \rightarrow \infty$. ***[refine]***

_refine_ls_goodness_of_fit_ref (*numb, su*)

The least-squares goodness-of-fit parameter S for all reflections included in the refinement after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also **_refine_ls_restrained_S_** definitions.

$$S = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{N_{\rm ref} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine_ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight $(1/u^2)$, u = the standard uncertainty, $N_{\rm ref}$ = the number of reflections used in the refinement, $N_{\rm param}$ = the number of refined parameters; the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_hydrogen_treatment (*char*)

Treatment of hydrogen atoms in the least-squares refinement.

The data value must be one of the following:

refall	refined all H-atom parameters
refxyz	refined H-atom coordinates only
refU	refined H-atom U's only
noref	no refinement of H-atom parameters
constr	H-atom parameters constrained
mixed	some constrained, some independent
undef	H-atom parameters not defined

Where no value is given, the assumed value is 'undef'. **[refine]**

_refine_ls_matrix_type (*char*)

Type of matrix used to accumulate the least-squares derivatives.

The data value must be one of the following:

full	full
fullcycle	full with fixed elements per cycle
atomblock	block diagonal per atom
userblock	user-defined blocks
diagonal	diagonal elements only
sparse	selected elements only

Where no value is given, the assumed value is 'full'. **[refine]**

_refine_ls_number_constraints (*numb*)

The number of constrained (non-refined or dependent) parameters in the least-squares process. These may be due to symmetry or any other constraint process (*e.g.* rigid-body refinement). See also **_atom_site_constraints** and **_atom_site_refinement_flags**. A general description of constraints may appear in **_refine_special_details**.

The permitted range is $0 \rightarrow \infty$. Where no value is given, the assumed value is '0'. **[refine]**

_refine_ls_number_parameters (*numb*)

The number of parameters refined in the least-squares process. If possible, this number should include some contribution from the restrained parameters. The restrained parameters are distinct from the constrained parameters (where one or more parameters are linearly dependent on the refined value of another). Least-squares restraints often depend on geometry or energy considerations and this makes their direct contribution to this number, and to the goodness-of-fit calculation, difficult to assess.

The permitted range is $0 \rightarrow \infty$. **[refine]**

_refine_ls_number_reflns (*numb*)

The number of unique reflections contributing to the least-squares refinement calculation.

The permitted range is $0 \rightarrow \infty$. **[refine]**

_refine_ls_number_restraints (*numb*)

The number of restrained parameters. These are parameters which are not directly dependent on another refined parameter. Restrained parameters often involve geometry or energy dependencies. See also **_atom_site_constraints** and **_atom_site_refinement_flags**. A general description of refinement constraints may appear in **_refine_special_details**.

The permitted range is $0 \rightarrow \infty$. **[refine]**

_refine_ls_R_factor_all (*numb*)

Residual factor for all reflections satisfying the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. This is the conventional R factor. See also **_refine_ls_wR_factor_** definitions.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_R_factor_gt (*numb*)

Residual factor for the reflections (with number given by **_reflns_number_gt**) judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns_threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. This is the conventional *R* factor. See also **_refine_ls_wR_factor_** definitions.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_R_factor_obs** (alternate). **[refine]**

_refine_ls_R_factor_obs (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refine_ls_R_factor_gt.

*Residual factor for the reflections classified as 'observed' (see **_reflns_observed_criterion**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. This is the conventional R factor. See also **_refine_ls_wR_factor_** definitions.*

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. ***[refine]***

_refine_ls_R_Fsqd_factor (*numb*)

Residual factor $R(F^2)$, calculated on the squared amplitudes of the observed and calculated structure factors, for significantly intense reflections (satisfying **_reflns_threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**.

$$R(F^2) = \frac{\sum |F^2_{\rm obs} - F^2_{\rm calc}|}{\sum |F^2_{\rm obs}|},$$

where $F^2_{\rm obs}$ = squares of the observed structure-factor amplitudes, $F^2_{\rm calc}$ = squares of the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_R_I_factor (*numb*)

Residual factor $R(I)$ for significantly intense reflections (satisfying **_reflns_threshold_expression**) and included in the refinement. This is most often calculated in Rietveld refinements against powder data, where it is referred to as R_B or $R_{\rm Bragg}$.

$$R(I) = \frac{\sum |I_{\rm obs} - I_{\rm calc}|}{\sum |I_{\rm obs}|},$$

where $I_{\rm obs}$ = the net observed intensities, $I_{\rm calc}$ = the net calculated intensities and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_restrained_S_all (*numb*)

The least-squares goodness-of-fit parameter S' for all reflections after the final cycle of least-squares refinement. This parameter explicitly includes the restraints applied in the least-squares process. See also **_refine_ls_goodness_of_fit_** definitions.

$$S' = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right| + \sum_r \left| w_r |P_{\rm calc} - P_{\rm targ}|^2 \right|}{N_{\rm ref} + N_{\rm restr} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine_ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight [1/square of standard uncertainty (e.s.d.)], $P_{\rm calc}$ = the calculated restraint values, $P_{\rm targ}$ = the target restraint values, w_r = the restraint weight, $N_{\rm ref}$ = the number of reflections used in the refinement (see **_refine_ls_number_reflns_obs**), $N_{\rm restr}$ = the number of restraints (see **_refine_ls_number_restraints**) and $N_{\rm param}$ = the number of refined parameters (see **_refine_ls_number_parameters**); the sum $\sum$ is taken over the specified reflections and the sum $\sum_r$ is taken over the restraints.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_restrained_S_gt (*numb*)

The least-squares goodness-of-fit parameter S' for significantly intense reflections (satisfying **_reflns_threshold_expression**) after the final cycle of least-squares refinement. This parameter explicitly includes the restraints applied in the least-squares process. See also **_refine_ls_goodness_of_fit_** definitions.

$$S' = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right| + \sum_r \left| w_r |P_{\rm calc} - P_{\rm targ}|^2 \right|}{N_{\rm ref} + N_{\rm restr} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine_ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight [1/square of standard uncertainty (e.s.d.)], $P_{\rm calc}$ = the calculated restraint values, $P_{\rm targ}$ = the target restraint values, w_r = the restraint weight, $N_{\rm ref}$ = the number of reflections used in the refinement (see **_refine_ls_number_reflns_obs**), $N_{\rm restr}$ = the number of restraints (see **_refine_ls_number_restraints**) and $N_{\rm param}$ = the number of refined parameters (see **_refine_ls_number_parameters**); the sum $\sum$ is taken over the specified reflections and the sum $\sum_r$ is taken over the restraints.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_restrained_S_obs** (alternate). **[refine]**

_refine_ls_restrained_S_obs (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refine_ls_restrained_S_gt.

*The least-squares goodness-of-fit parameter S' for observed reflections after the final cycle of least-squares refinement. This parameter explicitly includes the restraints applied in the least-squares process. See also **_refine_ls_goodness_of_fit_** definitions.*

$$S' = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right| + \sum_r \left| w_r |P_{\rm calc} - P_{\rm targ}|^2 \right|}{N_{\rm ref} + N_{\rm restr} - N_{\rm param}} \right)^{1/2},$$

*where $Y_{\rm obs}$ = the observed coefficients (see **_refine_ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine_ls_structure_factor_coef**), w = the least-squares reflection weight [1/square of standard uncertainty (e.s.d.)], $P_{\rm calc}$ = the calculated restraint values, $P_{\rm targ}$ = the target restraint values, w_r = the restraint weight, $N_{\rm ref}$ = the number of reflections used in the refinement (see **_refine_ls_number_reflns_obs**), $N_{\rm restr}$ = the number of restraints (see **_refine_ls_number_restraints**) and $N_{\rm param}$ = the number of refined parameters (see **_refine_ls_number_parameters**); the sum $\sum$ is taken over the specified reflections and the sum $\sum_r$ is taken over the restraints.*

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_shift/esd_max (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refine_ls_shift/su_max.

The largest ratio of the final least-squares parameter shift to the final standard uncertainty (s.u., formerly described as estimated standard deviation, e.s.d.).

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_shift/esd_mean (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refine_ls_shift/su_mean.

The average ratio of the final least-squares parameter shift to the final standard uncertainty (s.u., formerly described as estimated standard deviation, e.s.d.).

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_shift/su_max (*numb*)

The largest ratio of the final least-squares parameter shift to the final standard uncertainty.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_shift/esd_max** (alternate). **[refine]**

_refine_ls_shift/su_max_lt (*numb*)

An upper limit for the largest ratio of the final least-squares parameter shift to the final standard uncertainty. This item is used when the largest value of the shift divided by the final standard uncertainty is too small to measure.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_shift/su_max** (alternate). **[refine]**

_refine_ls_shift/su_mean (*numb*)

The average ratio of the final least-squares parameter shift to the final standard uncertainty.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_shift/esd_mean** (alternate). **[refine]**

_refine_ls_shift/su_mean_lt (*numb*)

An upper limit for the average ratio of the final least-squares parameter shift to the final standard uncertainty. This item is used when the average value of the shift divided by the final standard uncertainty is too small to measure.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_shift/su_mean** (alternate). **[refine]**

_refine_ls_structure_factor_coef (*char*)

Structure-factor coefficient $|F|$, F^2 or I used in the least-squares refinement process.

The data value must be one of the following:

F	structure-factor magnitude
Fsqd	structure factor squared
Inet	net intensity

Where no value is given, the assumed value is 'F'. **[refine]**

_refine_ls_weighting_details (*char*)

A description of special aspects of the weighting scheme used in the least-squares refinement. Used to describe the weighting when the value of **_refine_ls_weighting_scheme** is specified as 'calc'.

Example:

```
; Sigdel model of Konnert-Hendrickson:
  Sigdel = Afsig +  Bfsig*(sin(\q)/\l - 1/6)
  Afsig = 22.0, Bfsig = 150.0 at the beginning of refinement.
  Afsig = 16.0, Bfsig =  60.0 at the end of refinement.
;
```

[refine]

_refine_ls_weighting_scheme (*char*)

The weighting scheme applied in the least-squares process. The standard code may be followed by a description of the weight (but see **_refine_ls_weighting_details** for a preferred approach).

The data value must be one of the following:

sigma	based on measured s.u.'s
unit	unit or no weights applied
calc	calculated weights applied

Where no value is given, the assumed value is 'sigma'. **[refine]**

_refine_ls_wR_factor_all (*numb*)

Weighted residual factors for all reflections. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. See also the **_refine_ls_R_factor_** definitions.

$$wR = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |w Y_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine_ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine_ls_structure_factor_coef**, w = the least-squares weight and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_ls_wR_factor_gt (*numb*)

Weighted residual factors for significantly intense reflections (satisfying **_reflns_threshold_expression**) included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. See also the **_refine_ls_R_factor_** definitions.

$$wR = \left(\frac{\sum \left| w|Y_{obs} - Y_{calc}|^2 \right|}{\sum |wY_{obs}^2|} \right)^{1/2},$$

where Y_{obs} = the observed amplitude specified by **_refine_ls_structure_factor_coef**, Y_{calc} = the calculated amplitude specified by **_refine_ls_structure_factor_coef**, w = the least-squares weight and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_refine_ls_wR_factor_obs** (alternate). **[refine]**

_refine_ls_wR_factor_obs (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refine_ls_wR_factor_gt.

*Weighted residual factors for the reflections classified as 'observed' (see **_reflns_observed_criterion**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. See also the **_refine_ls_R_factor_** definitions.*

$$wR = \left(\frac{\sum \left| w|Y_{obs} - Y_{calc}|^2 \right|}{\sum |wY_{obs}^2|} \right)^{1/2},$$

*where Y_{obs} = the observed amplitude specified by **_refine_ls_structure_factor_coef**, Y_{calc} = the calculated amplitude specified by **_refine_ls_structure_factor_coef**, w = the least-squares weight and the sum is taken over the specified reflections.*

The permitted range is $0.0 \rightarrow \infty$. ***[refine]***

_refine_ls_wR_factor_ref (*numb*)

Weighted residual factors for all reflections included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_d_res_high** and **_refine_ls_d_res_low**. See also the **_refine_ls_R_factor_** definitions.

$$wR = \left(\frac{\sum \left| w|Y_{obs} - Y_{calc}|^2 \right|}{\sum |wY_{obs}^2|} \right)^{1/2},$$

where Y_{obs} = the observed amplitude specified by **_refine_ls_structure_factor_coef**, Y_{calc} = the calculated amplitude specified by **_refine_ls_structure_factor_coef**, w = the least-squares weight and the sum is taken over the specified reflections.

The permitted range is $0.0 \rightarrow \infty$. **[refine]**

_refine_special_details (*char*)

Description of special aspects of the refinement process.

[refine]

REFINE_LS_CLASS

Data items in the REFINE_LS_CLASS category record details (for each reflection class separately) about the reflections used for the structure refinement.

*Example 1 – example for a modulated structure extracted from van Smaalen [J. Phys. Condens. Matter (1991), **3**, 1247–1263.]*

```
loop_
    _refine_ls_class_R_factor_gt
    _refine_ls_class_code
       0.057    'Main'
       0.074    'Com'
       0.064    'NbRefls'
       0.046    'LaRefls'
       0.112    'Sat1'
       0.177    'Sat2'
```

_refine_ls_class_code (*char*)

The code identifying a certain reflection class. This code must match a **_reflns_class_code**.

Appears in list. **Must** match parent data name **_reflns_class_code**.

Examples: '1', 'm1', 's2'. **[refine_ls_class]**

_refine_ls_class_d_res_high (*numb*)

For each reflection class, the highest resolution in ångströms for the reflections used in the refinement. This is the lowest d value in a reflection class.

Appears in list containing **_refine_ls_class_code**.

The permitted range is $0.0 \rightarrow \infty$. **[refine_ls_class]**

_refine_ls_class_d_res_low (*numb*)

For each reflection class, the lowest resolution in ångströms for the reflections used in the refinement. This is the highest d value in a reflection class.

Appears in list containing **_refine_ls_class_code**.

The permitted range is $0.0 \rightarrow \infty$. **[refine_ls_class]**

_refine_ls_class_R_factor_all
_refine_ls_class_R_factor_gt (*numb*)

For each reflection class, the residual factors for all reflections, and for significantly intense reflections (see **_reflns_threshold_expression**), included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_class_d_res_high** and **_refine_ls_class_d_res_low**. This is the conventional R factor.

$$R = \frac{\sum |F_{obs} - F_{calc}|}{\sum |F_{obs}|},$$

where F_{obs} = the observed structure-factor amplitudes, F_{calc} = the calculated structure-factor amplitudes and the sum is taken over the specified reflections. See also **_refine_ls_class_wR_factor_all** definitions.

Appears in list containing **_refine_ls_class_code**.

The permitted range is $0.0 \rightarrow \infty$. **[refine_ls_class]**

_refine_ls_class_R_Fsqd_factor (*numb*)

For each reflection class, the residual factor $R(F^2)$ calculated on the squared amplitudes of the observed and calculated structure factors for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns_threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_class_d_res_high** and **_refine_ls_class_d_res_low**.

$$R(F^2) = \frac{\sum |F_{obs}^2 - F_{calc}^2|}{\sum |F_{obs}^2|},$$

where F_{obs}^2 = squares of the observed structure-factor amplitudes, F_{calc}^2 = squares of the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

Appears in list containing **_refine_ls_class_code**.

The permitted range is $0.0 \rightarrow \infty$. **[refine_ls_class]**

_refine_ls_class_R_I_factor (*numb*)
For each reflection class, the residual factor $R(I)$ for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns_threshold_expression**) and included in the refinement. This is most often calculated in Rietveld refinements against powder data, where it is referred to as R_B or $R_{\rm Bragg}$.

$$R(I) = \frac{\sum |I_{\rm obs} - I_{\rm calc}|}{\sum |I_{\rm obs}|},$$

where $I_{\rm obs}$ = the net observed intensities, $I_{\rm calc}$ = the net calculated intensities and the sum is taken over the specified reflections.
Appears in list containing **_refine_ls_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[refine_ls_class]**

_refine_ls_class_wR_factor_all (*numb*)
For each reflection class, the weighted residual factors for all reflections included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_class_d_res_high** and **_refine_ls_class_d_res_low**.

$$wR = \left(\frac{\sum \left| w |Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitudes specified by **_refine_ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitudes specified by **_refine_ls_structure_factor_coef**, w = the least-squares weights and the sum is taken over the reflections of this class. See also **_refine_ls_class_R_factor_** definitions.
Appears in list containing **_refine_ls_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[refine_ls_class]**

REFLN

Data items in the REFLN category record details about the reflections used to determine the ATOM_SITE data items. The REFLN data items refer to individual reflections and must be included in looped lists. The REFLNS data items specify the parameters that apply to all reflections. The REFLNS data items are not looped.

*Example 1 – based on data set fetod of Todres, Yanovsky, Ermekov & Struchkov [Acta Cryst. (1993), C**49**, 1352–1354].*

```
loop_
 _refln_index_h
 _refln_index_k
 _refln_index_l
 _refln_F_squared_calc
 _refln_F_squared_meas
 _refln_F_squared_sigma
 _refln_include_status
   2   0   0       85.57       58.90       1.45 o
   3   0   0    15718.18    15631.06      30.40 o
   4   0   0    55613.11    49840.09      61.86 o
   5   0   0      246.85      241.86      10.02 o
   6   0   0       82.16       69.97       1.93 o
   7   0   0     1133.62      947.79      11.78 o
   8   0   0     2558.04     2453.33      20.44 o
   9   0   0      283.88      393.66       7.79 o
  10   0   0      283.70      171.98       4.26 o
```

Example 2 – based on standard test data set p6122 of the Xtal distribution [Hall, King & Stewart (1995). Xtal3.4 User's Manual. University of Western Australia].

```
loop_
_refln_index_h
_refln_index_k
_refln_index_l
_refln_F_meas
_refln_F_calc
_refln_F_sigma
_refln_include_status
_refln_scale_group_code
    0   0   6  34.935  36.034   3.143  o  1
    0   0  12  42.599  40.855   2.131  o  1
    0   1   0  42.500  42.507   4.719  o  1
    0   1   1  59.172  57.976   4.719  o  1
    0   1   2  89.694  94.741   4.325  o  1
    0   1   3  51.743  52.241   3.850  o  1
    0   1   4   9.294  10.318   2.346  o  1
    0   1   5  41.160  39.951   3.313  o  1
    0   1   6   6.755   7.102    .895  <  1
    0   1   7  30.693  31.171   2.668  o  1
    0   1   8  12.324  12.085   2.391  o  1
    0   1   9  15.348  15.122   2.239  o  1
    0   1  10  17.622  19.605   1.997  o  1
```

_refln_A_calc
_refln_A_meas (*numb*)
The calculated and measured structure-factor component A (in electrons for X-ray diffraction).

$$A = |F| \cos(\text{phase}).$$

Appears in list containing **_refln_index_**. **[refln]**

_refln_B_calc
_refln_B_meas (*numb*)
The calculated and measured structure-factor component B (in electrons for X-ray diffraction).

$$B = |F| \sin(\text{phase}).$$

Appears in list containing **_refln_index_**. **[refln]**

_refln_class_code (*char*)
The code identifying the class to which this reflection has been assigned. This code must match a value of **_reflns_class_code**. Reflections may be grouped into classes for a variety of purposes. For example, for modulated structures each reflection class may be defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal lattice.
Appears in list containing **_refln_index_**. **Must** match parent data name **_reflns_class_code**. **[refln]**

_refln_crystal_id (*char*)
Code identifying each crystal if multiple crystals are used. Is used to link with **_exptl_crystal_id** in the **_exptl_crystal_** list.
Appears in list containing **_refln_index_**. **Must** match parent data name **_exptl_crystal_id**. **[refln]**

_refln_d_spacing (*numb*)
The d spacing in ångströms for this reflection. This is related to the $(\sin\theta)/\lambda$ value by the expression **_refln_d_spacing** = 2/(**_refln_sint/lambda**).
Appears in list containing **_refln_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[refln]**

_refln_F_calc
_refln_F_meas
_refln_F_sigma (*numb*)

The calculated, measured and standard uncertainty (derived from measurement) of the structure factors (in electrons for X-ray diffraction).

Appears in list containing **_refln_index_**. **[refln]**

_refln_F_squared_calc
_refln_F_squared_meas
_refln_F_squared_sigma (*numb*)

Calculated, measured and estimated standard uncertainty (derived from measurement) of the squared structure factors (in electrons squared for X-ray diffraction).

Appears in list containing **_refln_index_**. **[refln]**

_refln_include_status (*char*)

Classification of a reflection indicating its status with respect to inclusion in the refinement and the calculation of *R* factors.

Appears in list containing **_refln_index_**.

Related item: **_refln_observed_status** (alternate).

The data value must be one of the following:

o	(lower-case letter o for 'observed') satisfies **_refine_ls_d_res_high**, satisfies **_refine_ls_d_res_low** and exceeds **_reflns_threshold_expression**
<	satisfies **_refine_ls_d_res_high**, satisfies **_refine_ls_d_res_low** and does not exceed **_reflns_threshold_ expression**
-	systematically absent reflection
x	unreliable measurement – not used
h	does not satisfy **_refine_ls_d_res_high**
l	does not satisfy **_refine_ls_d_res_low**

Where no value is given, the assumed value is 'o'. **[refln]**

_refln_index_h
_refln_index_k
_refln_index_l (*numb*)

Miller indices of the reflection. The values of the Miller indices in the REFLN category must correspond to the cell defined by the cell lengths and cell angles in the CELL category.

Appears in list as essential element of loop structure. **[refln]**

_refln_intensity_calc
_refln_intensity_meas
_refln_intensity_sigma (*numb*)

The calculated, measured and standard uncertainty (derived from measurement) of the intensity, all in the same arbitrary units as **_refln_intensity_meas**.

Appears in list containing **_refln_index_**. **[refln]**

_refln_mean_path_length_tbar (*numb*)

Mean path length in millimetres through the crystal for this reflection.

Appears in list containing **_refln_index_**.

The permitted range is $0.0 \rightarrow \infty$. **[refln]**

_refln_observed_status (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _refln_include_status.

Classification of a reflection indicating its status with respect to inclusion in the refinement and the calculation of R factors.

*Appears in list containing **_refln_index_**.*

The data value must be one of the following:

o	*satisfies **_refine_ls_d_res_high**, satisfies **_refine_ls_d_res_low** and observed by **_reflns_observed_criterion***
<	*satisfies **_refine_ls_d_res_high**, satisfies **_refine_ls_d_res_low** and unobserved by **_reflns_observed_criterion***
-	*systematically absent reflection*
x	*unreliable measurement – not used*
h	*does not satisfy **_refine_ls_d_res_high***
l	*does not satisfy **_refine_ls_d_res_low***

Where no value is given, the assumed value is 'o'. ***[refln]***

_refln_phase_calc (*numb*)

The calculated structure-factor phase in degrees.

Appears in list containing **_refln_index_**. **[refln]**

_refln_phase_meas (*numb, su*)

The measured structure-factor phase in degrees.

Appears in list containing **_refln_index_**. **[refln]**

_refln_refinement_status (*char*)

Status of a reflection in the structure-refinement process.

Appears in list containing **_refln_index_**.

The data value must be one of the following:

incl	included in least-squares process
excl	excluded from least-squares process
extn	excluded due to extinction

Where no value is given, the assumed value is 'incl'. **[refln]**

_refln_scale_group_code (*char*)

Code identifying the structure-factor scale. This code must correspond to one of the **_reflns_scale_group_code** values.

Appears in list containing **_refln_index_**. **Must** match parent data name **_reflns_scale_group_code**.

Examples: '1', '2', '3', 's1', 'A', 'B', 'c1', 'c2', 'c3'. **[refln]**

_refln_sint/lambda (*numb*)

The $(\sin\theta)/\lambda$ value in reciprocal ångströms for this reflection.

Appears in list containing **_refln_index_**.

The permitted range is $0.0 \rightarrow \infty$. **[refln]**

_refln_symmetry_epsilon (*numb*)

The symmetry reinforcement factor corresponding to the number of times the reflection indices are generated identically from the space-group symmetry operations.

Appears in list containing **_refln_index_**.

The permitted range is $1 \rightarrow 48$. **[refln]**

_refln_symmetry_multiplicity (*numb*)

The number of reflections symmetry-equivalent under the Laue symmetry to the present reflection. In the Laue symmetry, Friedel opposites (*hkl* and $-h\,-k\,-l$) are equivalent. Tables of symmetry-equivalent reflections are available in *International Tables for Crystallography* Volume A (2002), Chapter 10.1.

Appears in list containing **_refln_index_**.

The permitted range is $1 \rightarrow 48$. **[refln]**

_refln_wavelength (*numb*)
The mean wavelength in ångströms of the radiation used to measure this reflection. This is an important parameter for data collected using energy-dispersive detectors or the Laue method.
Appears in list containing **_refln_index_**.
The permitted range is $0.0 \rightarrow \infty$. **[refln]**

_refln_wavelength_id (*char*)
Code identifying the wavelength in the **_diffrn_radiation_** list. See **_diffrn_radiation_wavelength_id**.
Appears in list containing **_refln_index_**. **Must** match parent data name **_diffrn_radiation_wavelength_id**. **[refln]**

REFLNS

Data items in the REFLNS category record details about the reflections used to determine the ATOM_SITE data items.

The REFLN data items refer to individual reflections and must be included in looped lists.

The REFLNS data items specify the parameters that apply to all reflections. The REFLNS data items are not looped.

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_reflns_limit_h_min              0
_reflns_limit_h_max              6
_reflns_limit_k_min              0
_reflns_limit_k_max              17
_reflns_limit_l_min              0
_reflns_limit_l_max              22
_reflns_number_total             1592
_reflns_number_gt                1408
_reflns_threshold_expression     'F > 6.0u(F)'
_reflns_d_resolution_high        0.8733
_reflns_d_resolution_low         11.9202
```

_reflns_d_resolution_high
_reflns_d_resolution_low (*numb*)
The highest and lowest resolution in ångströms for the reflections. These are the smallest and largest d values.
The permitted range is $0.0 \rightarrow \infty$. **[reflns]**

_reflns_Friedel_coverage (*numb*)
The proportion of Friedel-related reflections present in the number of 'independent' reflections specified by the item **_reflns_number_total**. This proportion is calculated as the ratio

$$\frac{[N(\text{crystal class}) - N(\text{Laue symmetry})]}{N(\text{Laue symmetry})},$$

where, working from the **_diffrn_refln_** list, N(crystal class) is the number of reflections obtained on averaging under the symmetry of the crystal class and N(Laue symmetry) is the number of reflections obtained on averaging under the Laue symmetry.

Examples: (*a*) For centrosymmetric structures, **_reflns_Friedel_coverage** is necessarily equal to 0.0 as the crystal class is identical to the Laue symmetry. (*b*) For whole-sphere data for a crystal in the space group $P1$, **_reflns_Friedel_coverage** is equal to 1.0, as no reflection hkl is equivalent to $-h\ -k\ -l$ in the crystal class and all Friedel pairs $\{hkl;\ -h\ -k\ -l\}$ have been measured. (*c*) For whole-sphere data in space group $Pmm2$, **_reflns_Friedel_coverage** will be < 1.0 because although reflections hkl and $-h\ -k\ -l$ are not equivalent when hkl indices are nonzero, they are when $l = 0$. (*d*) For a crystal in the space group $Pmm2$, measurements of the two inequivalent octants $h \geq 0, k \geq 0, l$ lead to the same value as in (*c*), whereas measurements of the two equivalent octants $h \geq 0, k, l \geq 0$ will lead to a value of zero for **_reflns_Friedel_coverage**.
The permitted range is $0.0 \rightarrow 1.0$. **[reflns]**

_reflns_limit_h_max
_reflns_limit_h_min
_reflns_limit_k_max
_reflns_limit_k_min
_reflns_limit_l_max
_reflns_limit_l_min (*numb*)
Miller indices limits for the reported reflections. These need not be the same as the **_diffrn_reflns_limit_** values.
[reflns]

_reflns_number_gt (*numb*)
The number of reflections in the **_refln_** list (not the **_diffrn_refln_** list) that are significantly intense, satisfying the criterion specified by **_reflns_threshold_expression**. This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Special characteristics of the reflections included in the **_refln_** list should be given in the item **_reflns_special_details**.
The permitted range is $0 \rightarrow \infty$.
Related item: **_reflns_number_observed** (alternate). **[reflns]**

_reflns_number_observed (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_number_gt.
*The number of 'observed' reflections in the **_refln_** list (not the **_diffrn_refln_** list). The observed reflections satisfy the threshold criterion specified by **_reflns_threshold_expression** (or the deprecated item **_reflns_observed_criterion**). They may include Friedel-equivalent reflections according to the nature of the structure and the procedures used. Special characteristics of the reflections included in the **_refln_** list should be given in the item **_reflns_special_details**.*
The permitted range is $0 \rightarrow \infty$. ***[reflns]***

_reflns_number_total (*numb*)
The total number of reflections in the **_refln_** list (not the **_diffrn_refln_** list). This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Special characteristics of the reflections included in the **_refln_** list should be given in the item **_reflns_special_details**.
The permitted range is $0 \rightarrow \infty$. **[reflns]**

_reflns_observed_criterion (*char*)
This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_threshold_expression.
The criterion used to classify a reflection as 'observed'. This criterion is usually expressed in terms of a $\sigma(I)$ *or* $\sigma(F)$ *threshold.*
Example: 'I>2u(I)'. ***[reflns]***

_reflns_special_details (*char*)
Description of the properties of the reported reflection list that are not given in other data items. In particular, this should include information about the averaging (or not) of symmetry-equivalent reflections including Friedel pairs.
[reflns]

_reflns_threshold_expression (*char*)
The threshold, usually based on multiples of $u(I)$, $u(F^2)$ or $u(F)$, that serves to identify significantly intense reflections, the number of which is given by **_reflns_number_gt**. These reflections are used in the calculation of **_refine_ls_R_factor_gt**.
Related item: **_reflns_observed_criterion** (alternate).
Example: '`I>2u(I)`'. **[reflns]**

REFLNS_CLASS

Data items in the REFLNS_CLASS category record details, for each reflection class, about the reflections used to determine the structural parameters.

Example 1 – corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
     _reflns_class_number_gt
     _reflns_class_code
        584     'Main'
        226     'Sat1'
        50      'Sat2'
```

_reflns_class_code (*char*)
The code identifying a certain reflection class.
Appears in list. May match child data name(s): **_refln_class_code**, **_refine_ls_class_code**.
Examples: '`1`', '`m1`', '`s2`'. **[reflns_class]**

_reflns_class_d_res_high (*numb*)
For each reflection class, the highest resolution in ångströms for the reflections used in the refinement. This is the smallest d value.
Appears in list containing **_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_d_res_low (*numb*)
For each reflection class, the lowest resolution in ångströms for the reflections used in the refinement. This is the largest d value.
Appears in list containing **_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_description (*char*)
Description of each reflection class.
Appears in list containing **_reflns_class_code**.
Examples: '`m=1 first order satellites`', '`H0L0 common projection reflections`'. **[reflns_class]**

_reflns_class_number_gt (*numb*)
For each reflection class, the number of significantly intense reflections (see **_reflns_threshold_expression**) in the **_refln_** list (not the **_diffrn_refln_** list). This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Special characteristics of the reflections included in the **_refln_** list should be given in the item **_reflns_special_details**.
Appears in list containing **_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_number_total (*numb*)
For each reflection class, the total number of reflections in the **_refln_** list (not the **_diffrn_refln_** list). This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Special characteristics of the reflections included in the **_refln_** list should be given in the item **_reflns_special_details**.
Appears in list containing **_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_R_factor_all
_reflns_class_R_factor_gt (*numb*)
For each reflection class, the residual factors for all reflections, and for significantly intense reflections (see **_reflns_threshold_expression**), included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class_d_res_high** and **_reflns_class_d_res_low**. This is the conventional R factor.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the reflections of this class. See also **_reflns_class_wR_factor_all** definitions.
Appears in list containing **_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_R_Fsqd_factor (*numb*)
For each reflection class, the residual factor $R(F^2)$ calculated on the squared amplitudes of the observed and calculated structure factors, for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns_threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class_d_res_high** and **_reflns_class_d_res_low**.

$$R(F^2) = \frac{\sum |F^2_{\rm obs} - F^2_{\rm calc}|}{\sum |F^2_{\rm obs}|},$$

where $F^2_{\rm obs}$ = squares of the observed structure-factor amplitudes, $F^2_{\rm calc}$ = squares of the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.
Appears in list containing **_reflns_class_code**.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_R_I_factor (*numb*)

For each reflection class, the residual factor $R(I)$ for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns_threshold_expression**) and included in the refinement. This is most often calculated in Rietveld refinements against powder data, where it is referred to as R_B or R_{Bragg}.

$$R(I) = \frac{\sum |I_{\mathrm{obs}} - I_{\mathrm{calc}}|}{\sum |I_{\mathrm{obs}}|},$$

where I_{obs} = the net observed intensities, I_{calc} = the net calculated intensities and the sum is taken over the reflections of this class.

Appears in list containing **_reflns_class_code**.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

_reflns_class_wR_factor_all (*numb*)

For each reflection class, the weighted residual factors for all reflections included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class_d_res_high** and **_reflns_class_d_res_low**.

$$wR = \left(\frac{\sum \left| w |Y_{\mathrm{obs}} - Y_{\mathrm{calc}}|^2 \right|}{\sum |w Y_{\mathrm{obs}}^2|} \right)^{1/2},$$

where Y_{obs} = the observed amplitudes specified by **_refine_ls_structure_factor_coef**, Y_{calc} = the calculated amplitudes specified by **_refine_ls_structure_factor_coef**, w = the least-squares weights and the sum is taken over the reflections of this class. See also **_reflns_class_R_factor_** definitions.

Appears in list containing **_reflns_class_code**.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_class]**

REFLNS_SCALE

Data items in the REFLNS_SCALE category record details about the structure-factor scales. They are referenced from within the REFLN list through **_refln_scale_group_code**.

Example 1 – based on standard test data set p6122 of the Xtal distribution [Hall, King & Stewart (1995). Xtal3.4 User's Manual. University of Western Australia].

```
loop_
_reflns_scale_group_code
_reflns_scale_meas_F
  1  .895447
  2  .912743
```

_reflns_scale_group_code (*char*)

The code identifying a scale **_reflns_scale_meas_**. These are linked to the **_refln_** list by the **_refln_scale_group_code**. These codes need not correspond to those in the **_diffrn_scale_** list.

Appears in list as essential element of loop structure. May match child data name(s): **_refln_scale_group_code**. **[reflns_scale]**

_reflns_scale_meas_F
_reflns_scale_meas_F_squared
_reflns_scale_meas_intensity (*numb, su*)

Scales associated with **_reflns_scale_group_code**.

Appears in list containing **_reflns_scale_group_code**.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_scale]**

REFLNS_SHELL

Data items in the REFLNS_SHELL category record details about the reflections used to determine the ATOM_SITE data items, as broken down by shells of resolution.

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_reflns_shell_d_res_high
_reflns_shell_d_res_low
_reflns_shell_meanI_over_uI_gt
_reflns_shell_number_measured_gt
_reflns_shell_number_unique_gt
_reflns_shell_percent_possible_gt
_reflns_shell_Rmerge_F_gt
  31.38  3.82  69.8  9024  2540  96.8   1.98
   3.82  3.03  26.1  7413  2364  95.1   3.85
   3.03  2.65  10.5  5640  2123  86.2   6.37
   2.65  2.41   6.4  4322  1882  76.8   8.01
   2.41  2.23   4.3  3247  1714  70.4   9.86
   2.23  2.10   3.1  1140   812  33.3  13.99
```

_reflns_shell_d_res_high (*numb*)

The highest resolution in ångströms for the reflections in this shell. This is the smallest d value.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_d_res_low (*numb*)

The lowest resolution in ångströms for the reflections in this shell. This is the largest d value.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_meanI_over_sigI_all (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_meanI_over_uI_all.

The ratio of the mean of the intensities of all reflections in this shell to the mean of the standard uncertainties of the intensities of all reflections in the resolution shell.

Appears in list. ***[reflns_shell]***

_reflns_shell_meanI_over_sigI_gt (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_meanI_over_uI_gt.

*The ratio of the mean of the intensities of the significantly intense reflections (see **_reflns_threshold_expression**) in this shell to the mean of the standard uncertainties of the intensities of the significantly intense reflections in the resolution shell.*

Appears in list. ***[reflns_shell]***

_reflns_shell_meanI_over_sigI_obs (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_meanI_over_sigI_gt.

*The ratio of the mean of the intensities of the reflections classified as 'observed' (see **_reflns_observed_criterion**) in this shell to the mean of the standard uncertainties of the intensities of the 'observed' reflections in the resolution shell.*

Appears in list. ***[reflns_shell]***

_reflns_shell_meanI_over_uI_all (*numb*)
The ratio of the mean of the intensities of all reflections in this shell to the mean of the standard uncertainties of the intensities of all reflections in the resolution shell.
Appears in list.
Related item: **_reflns_shell_meanI_over_sigI_all** (alternate).
[reflns_shell]

_reflns_shell_meanI_over_uI_gt (*numb*)
The ratio of the mean of the intensities of the significantly intense reflections (see **_reflns_threshold_expression**) in this shell to the mean of the standard uncertainties of the intensities of the significantly intense reflections in the resolution shell.
Appears in list.
Related items:
_reflns_shell_meanI_over_sigI_gt (alternate),
_reflns_shell_meanI_over_sigI_obs (alternate). **[reflns_shell]**

_reflns_shell_number_measured_all (*numb*)
The total number of reflections measured for this resolution shell.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_number_measured_gt (*numb*)
The number of significantly intense reflections (see **_reflns_threshold_expression**) measured for this resolution shell.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_reflns_shell_number_measured_obs** (alternate).
[reflns_shell]

_reflns_shell_number_measured_obs (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_number_measured_gt.
*The number of reflections classified as 'observed' (see **_reflns_observed_criterion**) measured for this resolution shell.*
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. ***[reflns_shell]***

_reflns_shell_number_possible (*numb*)
The number of unique reflections it is possible to measure in this reflection shell.
Appears in list.
The permitted range is $0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_number_unique_all (*numb*)
The total number of measured reflections resulting from merging measured symmetry-equivalent reflections for this resolution shell.
Appears in list.
The permitted range is $0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_number_unique_gt (*numb*)
The total number of significantly intense reflections (see **_reflns_threshold_expression**) resulting from merging measured symmetry-equivalent reflections for this resolution shell.
Appears in list.
The permitted range is $0 \rightarrow \infty$.
Related item: **_reflns_shell_number_unique_obs** (alternate). **[reflns_shell]**

_reflns_shell_number_unique_obs (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_number_unique_gt.
*The total number of reflections classified as 'observed' (see **_reflns_observed_criterion**) resulting from merging measured symmetry-equivalent reflections for this resolution shell.*
Appears in list.
The permitted range is $0 \rightarrow \infty$. ***[reflns_shell]***

_reflns_shell_percent_possible_all (*numb*)
The percentage of geometrically possible reflections represented by all reflections measured for this resolution shell.
Appears in list.
The permitted range is $0.0 \rightarrow 100.0$. **[reflns_shell]**

_reflns_shell_percent_possible_gt (*numb*)
The percentage of geometrically possible reflections represented by significantly intense reflections (see **_reflns_threshold_expression**) measured for this resolution shell.
Appears in list.
The permitted range is $0.0 \rightarrow 100.0$.
Related item: **_reflns_shell_percent_possible_obs** (alternate).
[reflns_shell]

_reflns_shell_percent_possible_obs (*numb*)
This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_percent_possible_gt.
*The percentage of geometrically possible reflections represented by reflections classified as 'observed' (see **_reflns_observed_criterion**) measured for this resolution shell.*
Appears in list.
The permitted range is $0.0 \rightarrow 100.0$. ***[reflns_shell]***

_reflns_shell_Rmerge_F_all (*numb*)
The value of $R_{\rm merge}(F)$ for all reflections in a given shell.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F \rangle|)}{\sum_i(\sum_j \langle F \rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F \rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_Rmerge_F_gt (*numb*)
The value of $R_{\rm merge}(F)$ for significantly intense reflections (see **_reflns_threshold_expression**) in a given shell.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F \rangle|)}{\sum_i(\sum_j \langle F \rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F \rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$.
Related item: **_reflns_shell_Rmerge_F_obs** (alternate). **[reflns_shell]**

_reflns_shell_Rmerge_F_obs (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_Rmerge_F_gt.

*The value of $R_{\rm merge}(F)$ for reflections classified as 'observed' (see **_reflns_observed_criterion**) in a given shell.*

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F\rangle|)}{\sum_i(\sum_j \langle F\rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F\rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_Rmerge_I_all (*numb*)

The value of $R_{\rm merge}(I)$ for all reflections in a given shell.

$$R_{\rm merge}(I) = \frac{\sum_i(\sum_j |I_j - \langle I\rangle|)}{\sum_i(\sum_j \langle I\rangle)},$$

where I_j = the intensity of the jth observation of reflection i, $\langle I\rangle$ = the mean of the intensities of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

_reflns_shell_Rmerge_I_gt (*numb*)

The value of $R_{\rm merge}(I)$ for significantly intense reflections (see **_reflns_threshold_expression**) in a given shell.

$$R_{\rm merge}(I) = \frac{\sum_i(\sum_j |I_j - \langle I\rangle|)}{\sum_i(\sum_j \langle I\rangle)},$$

where I_j = the intensity of the jth observation of reflection i, $\langle I\rangle$ = the mean of the intensities of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$.

Related item: **_reflns_shell_Rmerge_I_obs** (alternate). **[reflns_shell]**

_reflns_shell_Rmerge_I_obs (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _reflns_shell_Rmerge_I_gt.

*The value of $R_{\rm merge}(I)$ for reflections classified as 'observed' (see **_reflns_observed_criterion**) in a given shell.*

$$R_{\rm merge}(I) = \frac{\sum_i(\sum_j |I_j - \langle I\rangle|)}{\sum_i(\sum_j \langle I\rangle)},$$

where I_j = the intensity of the jth observation of reflection i, $\langle I\rangle$ = the mean of the intensities of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$. **[reflns_shell]**

SPACE_GROUP

Contains all the data items that refer to the space group as a whole, such as its name or crystal system. They may be looped, for example, in a list of space groups and their properties.

Only a subset of the SPACE_GROUP category items appear in the core dictionary. The remainder are found in the symmetry CIF dictionary. Space-group types are identified by their number as given in *International Tables for Crystallography* Vol. A. Specific settings of the space groups can be identified either by their Hall symbol or by specifying their symmetry operations. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but several different Hermann–Mauguin symbols may refer to the same space-group type. A Hermann–Mauguin symbol contains information on the choice of the basis, but not on the choice of origin. Different formats for the Hermann–Mauguin symbol are found in the symmetry CIF dictionary.

Example 1 – the monoclinic space group No. 15 with unique axis b.

```
_space_group_id                 1
_space_group_name_H-M_alt       'C 2/c'
_space_group_IT_number          15
_space_group_name_Hall          '-C 2yc'
_space_group_crystal_system     monoclinic
```

_space_group_crystal_system (*char*)

The name of the system of geometric crystal classes of space groups (crystal system) to which the space group belongs. Note that rhombohedral space groups belong to the trigonal system.

May appear in list containing **_space_group_id**.

Related item: **_symmetry_cell_setting** (alternate).

The data value must be one of the following:

- triclinic
- monoclinic
- orthorhombic
- tetragonal
- trigonal
- hexagonal
- cubic

[space_group]

_space_group_id (*char*)

This is an identifier needed if **_space_group_** items are looped.

Appears in list as essential element of loop structure. May match child data name(s): **_space_group_symop_sg_id**. **[space_group]**

_space_group_IT_number (*numb*)

The number as assigned in *International Tables for Crystallography* Vol. A, specifying the proper affine class (*i.e.* the orientation-preserving affine class) of space groups (crystallographic space-group type) to which the space group belongs. This number defines the space-group type but not the coordinate system in which it is expressed.

May appear in list containing **_space_group_id**.

The permitted range is $1 \rightarrow 230$.

Related item: **_symmetry_Int_Tables_number** (alternate). **[space_group]**

_space_group_name_H-M_alt (*char*)

_space_group_name_H-M_alt allows any Hermann–Mauguin symbol to be given. The way in which this item is used is determined by the user and in general is not intended to be interpreted by computer. It may, for example, be used to give one of the extended Hermann–Mauguin symbols given in Table 4.3.2.1

of *International Tables for Crystallography* Vol. A (2002) or a Hermann–Mauguin symbol for a conventional or unconventional setting. Each component of the space-group name is separated by a space or an underscore. The use of a space is strongly recommended. The underscore is only retained because it was used in older files. It should not be used in new CIFs. Subscripts should appear without special symbols. Bars should be given as negative signs before the numbers to which they apply. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but a given space-group type may be described by more than one Hermann–Mauguin symbol. The space-group type is best described using **_space_group_IT_number**. The Hermann–Mauguin symbol may contain information on the choice of basis, but not on the choice of origin. To define the setting uniquely, use **_space_group_name_Hall** or list the symmetry operations.

May appear in list containing **_space_group_id**.

Related item: **_symmetry_space_group_name_H-M** (alternate).

Example:

```
; loop_
  _space_group_id
  _space_group_name_H-M_alt
  1    'C m c m'
  2    'C 2/c 2/m 21/m'
  3    'A m a m'
;  (three examples for space group No. 63)
```

[space_group]

_space_group_name_Hall (*char*)

Space-group symbol defined by Hall. Each component of the space-group name is separated by a space or an underscore. The use of a space is strongly recommended. The underscore is only retained because it was used in older files. It should not be used in new CIFs. **_space_group_name_Hall** uniquely defines the space group and its reference to a particular coordinate system.

References: Hall, S. R. (1981). *Acta Cryst.* A**37**, 517–525; erratum (1981), A**37**, 921. [See also *International Tables for Crystallography*, Vol. B (2001), Chapter 1.4, Appendix 1.4.2].

May appear in list containing **_space_group_id**.

Related item: **_symmetry_space_group_name_Hall** (alternate).

Examples: '`P 2c -2ac`' (equivalent to $Pca2_1$), '`-I 4bd 2ab 3`' (equivalent to $Ia\bar{3}d$).

[space_group]

SPACE_GROUP_SYMOP

Contains information about the symmetry operations of the space group.

Example 1 – the symmetry operations for the space group $P2_1/c$.

```
loop_
_space_group_symop_id
_space_group_symop_operation_xyz
  1    x,y,z
  2   -x,-y,-z
  3   -x,1/2+y,1/2-z
  4    x,1/2-y,1/2+z
```

_space_group_symop_id (*char*)

An arbitrary identifier that uniquely labels each symmetry operation in the list.

Appears in list as essential element of loop structure.

Related item: **_symmetry_equiv_pos_site_id** (alternate).

[space_group_symop]

_space_group_symop_operation_xyz (*char*)

A parsable string giving one of the symmetry operations of the space group in algebraic form. If W is a matrix representation of the rotational part of the symmetry operation defined by the positions and signs of x, y and z, and $\mathbf{w}$ is a column of translations defined by fractions, an equivalent position $\boldsymbol{x}'$ is generated from a given position $\boldsymbol{x}$ by

$$\boldsymbol{x}' = W\boldsymbol{x} + \mathbf{w}.$$

When a list of symmetry operations is given, it must contain a complete set of coordinate representatives which generates all the operations of the space group by the addition of all primitive translations of the space group. Such representatives are to be found as the coordinates of the general-equivalent position in *International Tables for Crystallography* Vol. A (2002), to which it is necessary to add any centring translations shown above the general-equivalent position. That is to say, it is necessary to list explicitly all the symmetry operations required to generate all the atoms in the unit cell defined by the setting used.

May appear in list containing **_space_group_symop_id**.

Related item: **_symmetry_equiv_pos_as_xyz** (alternate). Where no value is given, the assumed value is '`x,y,z`'.

Example: '`x,1/2-y,1/2+z`' (glide reflection through the plane (x, 1/4, z), with glide vector (1/2)**c**).

[space_group_symop]

_space_group_symop_sg_id (*numb*)

This must match a particular value of **_space_group_id**, allowing the symmetry operation to be identified with a particular space group.

May appear in list containing **_space_group_symop_id**. **Must** match parent data name **_space_group_id**.

[space_group_symop]

SYMMETRY

Data items in the SYMMETRY category record details about the space-group symmetry.

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_symmetry_cell_setting            orthorhombic
_symmetry_space_group_name_H-M    'P 21 21 21'
_symmetry_space_group_name_Hall   'P 2ac 2ab'
```

_symmetry_cell_setting (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _space_group_crystal_system.

The cell settings for this space-group symmetry.

The data value must be one of the following:

- triclinic
- monoclinic
- orthorhombic
- tetragonal
- rhombohedral
- trigonal
- hexagonal
- cubic

[symmetry]

_symmetry_Int_Tables_number (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _space_group_IT_number.

Space-group number from International Tables for Crystallography Vol. A (2002).

The permitted range is 1 → 230.

[symmetry]

_symmetry_space_group_name_H-M (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _space_group_name_H-M_alt.

Hermann–Mauguin space-group symbol. Note that the Hermann–Mauguin symbol does not necessarily contain complete information about the symmetry and the space-group origin. If used, always supply the full *symbol from International Tables for Crystallography Vol. A (2002) and indicate the origin and the setting if it is not implicit. If there is any doubt that the equivalent positions can be uniquely deduced from this symbol, specify the* ***_symmetry_equiv_pos_as_xyz*** *or* ****_Hall*** *data items as well. Leave spaces between symbols referring to different axes.*

Examples: '`P 1 21/m 1`', '`P 2/n 2/n 2/n (origin at -1)`', '`R -3 2/m`'.

[symmetry]

_symmetry_space_group_name_Hall (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _space_group_name_Hall.

Space-group symbol as described by Hall. This symbol gives the space-group setting explicitly. Leave spaces between the separate components of the symbol.

Reference: Hall, S. R. (1981). Acta Cryst. A ***37****, 517–525; erratum (1981), A* ***37****, 921.*

Examples: '`-P 2ac 2n`', '`-R 3 2"`', '`P 61 2 2 (0 0 -1)`'. ***[symmetry]***

SYMMETRY_EQUIV

Data items in the SYMMETRY_EQUIV category list the symmetry-equivalent positions for the space group.

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C ***47****, 2276–2277].*

```
loop_
_symmetry_equiv_pos_as_xyz
    +x,+y,+z  1/2-x,-y,1/2+z  1/2+x,1/2-y,-z  -x,1/2+y,1/2-z
```

Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C ***47****, 2276–2277].*

Formally, the value of **_symmetry_equiv_pos_site_id** can be any unique character string; it is recommended that it be assigned the sequence number of the list of equivalent positions for compatibility with older files in which it did not appear.

```
loop_
_symmetry_equiv_pos_site_id
_symmetry_equiv_pos_as_xyz
    1     x,y,z
    2     1/2-x,-y,1/2+z
    3     1/2+x,1/2-y,-z
    4     -x,1/2+y,1/2-z
```

_symmetry_equiv_pos_as_xyz (*char*)

This definition has been superseded and is retained here only for archival purposes. Use instead _space_group_symop_operation_xyz.

*Symmetry-equivalent position in the 'xyz' representation. Except for the space group P*1*, these data will be repeated in a loop. The format of the data item is as per International Tables for Crystallography Vol. A. (2002). All equivalent positions should be entered, including those for lattice centring and a centre of symmetry, if present.*

May appear in list.

Example: '`-y+x,-y,1/3+z`'. ***[symmetry_equiv]***

_symmetry_equiv_pos_site_id (*numb*)

This definition has been superseded and is retained here only for archival purposes. Use instead _space_group_symop_id.

A code identifying each entry in the ***_symmetry_equiv_pos_as_xyz*** *list. It is normally the sequence number of the entry in that list, and should be identified with the code 'n' in* ***_geom_*_symmetry_*** *codes of the form 'n_klm'.*

Appears in list containing ***_symmetry_equiv_pos_as_xyz****.* ***[symmetry_equiv]***

VALENCE_PARAM

Data items in the VALENCE_PARAM category define the parameters used for calculating bond valences from bond lengths. In addition to the parameters, a pointer is given to the reference (in VALENCE_REF) from which the bond-valence parameters were taken.

Example 1 – a bond-valence parameter list with accompanying references.

```
loop_
_valence_param_id
_valence_param_atom_1
_valence_param_atom_1_valence
_valence_param_atom_2
_valence_param_atom_2_valence
_valence_param_Ro
_valence_param_B
_valence_param_ref_id
_valence_param_details
  1   Cu 2 O -2 1.679 0.37 a .
  2   Cu 2 O -2 1.649 0.37 j .
  3   Cu 2 N -3 1.64  0.37 m '2-coordinate N'
  4   Cu 2 N -3 1.76  0.37 m '3-coordinate N'
loop_
_valence_ref_id
_valence_ref_reference
  a  'Brown & Altermatt (1985), Acta Cryst. B41, 244-247'
  j  'Liu & Thorp (1993), Inorg. Chem. 32, 4102-4205'
  m  'See, Krause & Strub (1998), Inorg. Chem. 37, 5369-5375'
```

_valence_param_atom_1 (*char*)

The element symbol of the first atom forming the bond whose bond-valence parameters are given in this category.

Appears in list containing **_valence_param_id**. **[valence_param]**

_valence_param_atom_1_valence (*numb*)

The valence (formal charge) of the first atom whose bond-valence parameters are given in this category.

Appears in list containing **_valence_param_id**. **[valence_param]**

_valence_param_atom_2 (*char*)

The element symbol of the second atom forming the bond whose bond-valence parameters are given in this category.

Appears in list containing **_valence_param_id**. **[valence_param]**

_valence_param_atom_2_valence (*numb*)

The valence (formal charge) of the second atom whose bond-valence parameters are given in this category.

Appears in list containing **_valence_param_id**. **[valence_param]**

_valence_param_B (*numb*)

The bond-valence parameter B used in the expression

$$s = \exp[(R_o - R)/B],$$

where s is the valence of a bond of length R.

Appears in list containing **_valence_param_id**. **[valence_param]**

_valence_param_details (*char*)

Details of or comments on the bond-valence parameters.

Appears in list containing **_valence_param_id**. **[valence_param]**

_valence_param_id (*char*)
An identifier for the valence parameters of a bond between the given atoms.
Appears in list. **[valence_param]**

_valence_param_ref_id (*char*)
An identifier which links to the reference to the source from which the bond-valence parameters are taken. A child of **_valence_ref_id**, which it must match.
Appears in list containing **_valence_param_id**. **Must** match parent data name **_valence_ref_id**. **[valence_param]**

_valence_param_Ro (*numb*)
The bond-valence parameter R_o used in the expression

$$s = \exp[(R_o - R)/B],$$

where s is the valence of a bond of length R.
Appears in list containing **_valence_param_id**. **[valence_param]**

VALENCE_REF

Data items in the VALENCE_REF category list the references from which the bond-valence parameters have been taken.

_valence_ref_id (*char*)
An identifier for items in this category. Parent of **_valence_param_ref_id**, which must have the same value.
Appears in list containing **_valence_ref_id**. May match child data name(s): **_valence_param_ref_id**. **[valence_ref]**

_valence_ref_reference (*char*)
Literature reference from which the valence parameters identified by **_valence_param_id** were taken.
Appears in list containing **_valence_ref_id**. **[valence_ref]**

4.2. Powder dictionary (pdCIF)

BY B. H. TOBY

This is version 1.0.1 of the powder CIF dictionary (pdCIF). The data names defined in this dictionary complement those in the core dictionary (Chapter 4.1) and should be used to describe the results of powder diffraction studies. The organization of powder data sets, especially for studies of multiphase samples and for studies referring to external calibration standards, is discussed in Chapter 3.3.

The pdCIF data structure departs from the rigorous relational nature of the core dictionary in that it does not adhere fully to the close coupling of data names and associated category names. The dictionary is therefore presented here strictly alphabetically by data name. Not all pdCIF categories are described in this dictionary. Further, the pdCIF dictionary defines items that belong to categories in the core CIF dictionary. Care must therefore be taken in checking the category membership of each data name; this is particularly important to ensure that items in the same category are presented together in the same looped lists. See Chapters 3.1 and 3.3 for a more complete discussion.

PD_BLOCK

`_pd_block_id` is used to assign a unique ID code to a data block. This code is then used for references between different blocks (see `_pd_block_diffractogram_id`, `_pd_calib_std_external_block_id` and `_pd_phase_block_id`). Note that a data block may contain only a single diffraction data set or information about a single crystalline phase. However, a single diffraction measurement may yield structural information on more than one phase, or a single structure determination may use more than one data set. Alternatively, results from a single data set, such as calibration parameters from measurements of a standard, may be used for many subsequent analyses. Through use of the ID code, a reference made between data sets may be preserved when the file is exported from the laboratory from which the CIF originated. The ID code assigned to each data block should be unique with respect to an ID code assigned for any other data block in the world. The naming scheme chosen for the block-ID format is designed to ensure uniqueness. It is the responsibility of a data archive site or local laboratory to create a catalogue of block IDs if that site wishes to resolve these references.

`_pd_block_diffractogram_id` (*char*)

A block ID code (see `_pd_block_id`) that identifies diffraction data contained in a data block other than the current block. This will occur most frequently when more than one set of diffraction data is used for a structure determination. The data block containing the diffraction data will contain a `_pd_block_id` code matching the code in `_pd_block_diffractogram_id`.

Appears in list. **[pd_proc]**

`_pd_block_id` (*char*)

Used to assign a unique character string to a block. Note that this code is not intended to be parsed; the concatenation of several strings is used in order to generate a string that can reasonably be expected to be unique.

This code is assigned by the originator of the data set and is used for references between different CIF blocks. The ID will normally be created when the block is first created. It is possible to loop more than one ID for a block: if changes or additions are made to the block later, a new ID may be assigned, but the original name should be retained.

The format for the ID code is

$$\langle \textit{date-time} \rangle | \langle \textit{block\_name} \rangle | \langle \textit{creator\_name} \rangle | \langle \textit{instr\_name} \rangle .$$

$\langle$*date-time*$\rangle$ is the date and time the CIF was created or modified. $\langle$*block_name*$\rangle$ is an arbitrary name assigned by the originator of the data set. It will usually match the name of the phase and possibly the name of the current CIF data block (*i.e.* the string *xxxx* in a `data_`*xxxx* identifier). It may be a sample name. $\langle$*creator_name*$\rangle$ is the name of the person who measured the diffractogram, or prepared or modified the CIF. $\langle$*instr_name*$\rangle$ is a unique name (as far as possible) for the data-collection instrument, preferably containing the instrument serial number for commercial instruments. It is also possible to use the Internet name or address for the instrument computer as a unique name.

As blocks are created in a CIF, the original sample identifier (*i.e.* $\langle$*block_name*$\rangle$) should be retained, but the $\langle$*creator_name*$\rangle$ may be changed and the $\langle$*date-time*$\rangle$ will always change. The $\langle$*date-time*$\rangle$ will usually match either the `_pd_meas_datetime_initiated` or the `_pd_proc_info_datetime` entry.

Within each section of the code, the following characters may be used:

`A-Z a-z 0-9 # & * . : , - _ + / ( ) \ [ ]`

The sections are separated with vertical rules '|', which are not allowed within the sections. Blank spaces may also not be used. Capitalization may be used within the ID code but should not be considered significant – searches for data-set ID names should be case-insensitive.

Date–time entries are in the standard CIF format '*yyyy-mm-dd*T*hh*:*mm*:*ss*+*zz*'. Use of seconds and a time zone is optional, but use of hours and minutes is strongly encouraged as this will help to ensure that the ID code is unique.

An archive site that wishes to make CIFs available *via* the web may substitute the URL for the file containing the appropriate block for the final two sections of the ID ($\langle$*creator_name*$\rangle$ and $\langle$*instr_name*$\rangle$). Note that this should not be done unless the archive site is prepared to keep the file available online indefinitely.

May appear in list.

Examples: '`1991-15-09T16:54|Si-std|B.Toby|D500#1234-987`', '`1991-15-09T16:54|SEPD7234|B.Toby|SEPD.IPNS.ANL.GOV`'. **[pd_block]**

PD_CALC

This section is used for storing a computed diffractogram trace. This may be a simulated powder pattern for a material from a program such as *LAZY/PULVERIX* or the computed intensities from a Rietveld refinement.

Affiliation: BRIAN H. TOBY, NIST Center for Neutron Research, National Institute of Standards and Technology, Gaithersburg, Maryland 20899-8562, USA.

_pd_calc_intensity_net
_pd_calc_intensity_total (*numb*)
Intensity values for a computed diffractogram at each angle setting. Values should be computed at the same locations as the processed diffractogram, and thus the numbers of points will be defined by **_pd_proc_number_of_points** and point positions may be defined using **_pd_proc_2theta_range_** or **_pd_proc_2theta_corrected**. Use **_pd_calc_intensity_net** if the computed diffractogram does not contain background or normalization corrections and thus is specified on the same scale as the **_pd_proc_intensity_net** values. Use **_pd_calc_intensity_total** if the computed diffraction pattern includes background or normalization corrections (or both) and thus is specified on the same scale as the observed intensities (**_pd_meas_counts_** or **_pd_meas_intensity_**). If an observed pattern is included, **_pd_calc_intensity_** should be looped with either **_pd_proc_intensity_net**, **_pd_meas_counts_** or **_pd_meas_intensity_**.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_calc_method (*char*)
A description of the method used for the calculation of the intensities in **_pd_calc_intensity_**. If the pattern was calculated from crystal structure data, the atom coordinates and other crystallographic information should be included using the core CIF **_atom_site_** and **_cell_** data items.
[pd_calc]

_pd_calc_point_id (*char*)
Arbitrary label identifying a calculated data point. Used to identify a specific entry in a list of values forming the calculated diffractogram. The role of this identifier may be adopted by **_pd_data_point_id** if measured, processed and calculated intensity values are combined in a single list.
Related item: **_pd_data_point_id** (alternate). **[pd_data]**

PD_CALIB

This section defines the parameters used for the calibration of the instrument that are used directly or indirectly in the interpretation of this data set. The information in this section of the CIF should generally be written when the intensities are first measured, but from then on should remain unchanged. Loops may be used for calibration information that differs by detector channel. The **_pd_calibration_** items, however, are never looped.

Example 1.

```
_pd_calib_std_external_block_id
        QuartzPlate|D500#1234-987|B.Toby|91-15-09|14:02
_pd_calib_std_external_name
        'Arkansas Stone quartz plate'
```

Example 2.
$2\theta_{\text{actual}} = 2\theta_{\text{setting}} + \arctan(\cos(P_1) / \{1/[P_0\,(CC - CH_0 - P_2\,CC^2)] - \sin(P_1)\})$. *This allows for the calibration of 2θ for a linear position-sensitive detector (PSD) where the PSD has been set so that the 'centre channel' (CH_0) is located at $2\theta_{\text{setting}}$ as a function of the channel number (CC). In addition to CH_0, variables P_0, P_1 and P_2 are calibration constants, where P_0 is the width of a PSD channel in degrees, P_1 is the angle of the PSD with respect to the perpendicular and P_2 is a quadratic term for nonlinearities in the detector.*

```
_pd_calibration_conversion_eqn
; 2\q~actual~ = 2\q~setting~ + arctan(
cos(P~1~)/{1/[P~0~ (CC - CH~0~ - P~2~ CC^2^)] - sin(P~1~)})
;
```

_pd_calib_detector_id (*char*)
A code which identifies the detector or channel number in a position-sensitive, energy-dispersive or other multiple-detector instrument. Note that this code should match the code name used for **_pd_meas_detector_id**.
Appears in list as essential element of loop structure. May match child data name(s): **_pd_meas_detector_id**. **[pd_calib]**

_pd_calib_detector_response (*numb*)
A value that indicates the relative sensitivity of each detector. This can compensate for differences in electronics, size and collimation. Usually, one detector or the mean for all detectors will be assigned the value of 1.
Appears in list containing **_pd_calib_detector_id**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_calib]**

_pd_calib_std_external_block_id
_pd_calib_std_external_name (*char*)
Identifies the data set used as an external standard for the diffraction angle or the intensity calibrations. ***_name** specifies the name of the material and ***_id** the **_pd_block_id** for the CIF containing calibration measurements. If more than one data set is used for calibration, these fields may be looped.
May appear in list containing **_pd_calib_detector_id**. **[pd_calib]**

_pd_calib_std_internal_mass_% (*numb, su*)
Per cent presence of the internal standard specified by the data item **_pd_calib_std_internal_name** expressed as 100 times the ratio of the amount of standard added to the original sample mass.
May appear in list containing **_pd_calib_detector_id**.
The permitted range is $0.0 \rightarrow 100.0$. **[pd_calib]**

_pd_calib_std_internal_name (*char*)
Identity of material(s) used as an internal intensity standard.
May appear in list containing **_pd_calib_detector_id**.
Examples: 'NIST 640a Silicon standard', 'Al2O3'. **[pd_calib]**

_pd_calib_2theta_offset
_pd_calib_2theta_off_point
_pd_calib_2theta_off_min
_pd_calib_2theta_off_max (*numb*)
_pd_calib_2theta_offset defines an offset angle (in degrees) used to calibrate 2θ (as defined in **_pd_meas_2theta_**). Calibration is done by adding the offset:

$$2\theta_{\text{calibrated}} = 2\theta_{\text{measured}} + 2\theta_{\text{offset}}.$$

For cases where the **_pd_calib_2theta_offset** value is not a constant, but rather varies with 2θ, a set of offset values can be supplied in a loop. In this case, the value where the offset has been determined can be specified as **_pd_calib_2theta_off_point**. Alternatively, a range where the offset is applicable can be specified using **_pd_calib_2theta_off_min** and **_pd_calib_2theta_off_max**.
May appear in list containing **_pd_calib_detector_id**.
The permitted range is $-180.0 \rightarrow 180.0$. **[pd_calib]**

_pd_calibration_conversion_eqn (*char*)
The calibration function for converting a channel number supplied in **_pd_meas_detector_id** for a position-sensitive or energy-dispersive detector or the distance supplied in **_pd_meas_position** to Q, energy, angle *etc.* Use **_pd_calib_std_external_** to define a pointer to the file or data block containing the information used to define this function.
Example:

```
; 2\q~actual~ = 2\q~setting~ + arctan(
cos(P~1~) / {1/[P~0~ (CC - CH~0~ - P~2~ CC^2^)] - sin(P~1~)})
;
```
[pd_calibration]

_pd_calibration_special_details (*char*)
Description of how the instrument was calibrated, particularly for instruments where calibration information is used to make hardware settings that would otherwise be invisible once data collection is completed. Do not use this item to specify information that can be specified using other **_pd_calib_** items.

[pd_calibration]

PD_CHAR

This section contains experimental (non-diffraction) information relevant to the chemical and physical nature of the material.

_pd_char_atten_coef_mu_obs
_pd_char_atten_coef_mu_calc (*numb*)
The observed and calculated linear attenuation coefficient, μ, in units of inverse millimetres. Note that this quantity is sometimes referred to as the mass absorption coefficient; however, this term accounts for other potentially significant losses of incident radiation, for example incoherent scattering of neutrons. The calculated μ will be obtained from the atomic content of the cell, the average density (allowing for specimen packing) and the radiation wavelength. The observed μ will be determined by a transmission measurement. Note that **_pd_char_atten_coef_mu_calc** will differ from **_exptl_absorpt_coefficient_mu** if the packing density is not unity.

The permitted range is $0.0 \rightarrow \infty$. **[pd_char]**

_pd_char_colour (*char*)
The colour of the material used for the measurement. To facilitate more standardized use of names, the following guidelines for colour naming developed by Peter Bayliss for the International Centre for Diffraction Data (ICDD) should be followed. Note that combinations of descriptors are separated by an underscore. Allowed colours are: colourless, white, black, gray, brown, red, pink, orange, yellow, green, blue, violet. Colours may be modified using prefixes of: light, dark, whitish, blackish, grayish, brownish, reddish, pinkish, orangish, yellowish, greenish, bluish. Intermediate hues may be indicated with two colours: *e.g.* blue_green or bluish_green. For metallic materials, the term metallic may be added: *e.g.* reddish_orange_metallic for copper. The ICDD standard allows commas to be used for minerals that occur with ranges of colours; however this usage is not appropriate for the description of a single sample.

Examples: 'dark_green', 'orange_red', 'brownish_red', 'yellow_metallic'.

[pd_char]

_pd_char_particle_morphology (*char*)
A description of the sample morphology and estimates for particle sizes (before grinding/sieving, if noted by **_pd_spec_preparation**). Include the method used for these estimates (SEM, visual estimate *etc.*).

[pd_char]

_pd_char_special_details (*char*)
Additional characterization information relevant to the sample or documentation of non-routine processing steps used for characterization.

[pd_char]

PD_DATA

The PD_DATA category contains raw, processed and calculated data points in a diffraction data set. In many cases, it is convenient to tabulate calculated values against the raw and processed measurements, and so the various **_pd_meas_**, **_pd_proc_** and **_pd_calc_** data items belonging to this category may be looped together. In some instances, however, it makes more sense to maintain separate tables of the data contributing to the measured and processed diffractograms (for example, a profile may be calculated at 2θ values different from those of the measured data points). To facilitate the identification of equivalent points in these separate tables, separate identifiers are defined.

Example 1 – data set collected for a two-phase sample (Al_2O_3/Si) by B. H. Toby, '1997-01-29T16:37|POWSET_01|B.Toby|..NIST-D5000'.

```
loop_
  _pd_data_point_id
  _pd_meas_intensity_total
  _pd_proc_ls_weight
  _pd_proc_intensity_bkg_calc
  _pd_calc_intensity_total
    1    240(15)     0.00417     214.5   214.5
    2    219(15)     0.00457     214.3   214.2
    3    206(14)     0.00485     214.0   214.0
    4    212(15)     0.00472     213.8   213.7
    5    190(14)     0.00526     213.5   213.5
    6    203(14)     0.00493     213.2   213.2
 # - - - - data truncated for brevity - - - -
```

Example 2 – the data of Example 1 split into three separate tables.

```
 loop_
  _pd_meas_point_id
  _pd_meas_intensity_total
    1  240(15) 2  219(15) 3  206(14) 4  212(15) 5  190(14)
    6  203(14)

 loop_
  _pd_proc_point_id
  _pd_proc_ls_weight
  _pd_proc_intensity_bkg_calc
    1  0.00417  214.5      2  0.00457  214.3
    3  0.00485  214.0      4  0.00472  213.8
    5  0.00526  213.5      6  0.00493  213.2

 loop_
  _pd_calc_point_id
  _pd_calc_intensity_total
    1   214.5 2   214.2 3   214.0 4   213.7 5   213.5
    6   213.2
```

Example 3 – hypothetical example where the measured and calculated points are not in one-to-one correspondence.

```
   loop_
_pd_meas_point_id
_pd_meas_2theta_scan
_pd_meas_intensity_total
       1  21.0   24
       2  21.2   32
       3  21.4   67
       4  21.6   98

   loop_
_pd_calc_point_id
_pd_proc_2theta_corrected
_pd_calc_intensity_total
       1   21.0  26
       1a  21.3  56
       4   21.6  76
       4a  21.9  90
```

_pd_data_point_id (*char*)
Arbitrary label identifying an entry in the table of diffractogram intensity values.

[pd_data]

PD_INSTR

This section contains information relevant to the instrument used for the diffraction measurement. For most laboratories, very little of this information will change, so a standard file may be prepared and included with each data set. Note that several definitions in the core CIF dictionary are relevant here. For example, use: **_diffrn_radiation_wavelength** for the source wavelength, **_diffrn_radiation_type** for the X-ray wavelength type, **_diffrn_source** for the radiation source, **_diffrn_radiation_polarisn_ratio** for the source polarization, **_diffrn_radiation_probe** for the radiation type. For data sets measured with partially monochromatized radiation, for example, where both $K\alpha_1$ and $K\alpha_2$ are present, it is important that all wavelengths present are included in a **loop_** using **_diffrn_radiation_wavelength** to define the wavelength and **_diffrn_radiation_wavelength_wt** to define the relative intensity of that wavelength. It is required that **_diffrn_radiation_wavelength_id** also be present in the wavelength loop. It may also be useful to create a 'dummy' ID to use for labelling peaks/reflections where the $K\alpha_1$ and $K\alpha_2$ wavelengths are not resolved. Set **_diffrn_radiation_wavelength_wt** to be 0 for such a dummy ID. In the **_pd_instr_** definitions, the term monochromator refers to a primary beam (pre-specimen) monochromator and the term analyser refers to post-diffraction (post-specimen) monochromator. The analyser may be fixed for a specific wavelength or may be capable of being scanned. For multiple-detector instruments it may be necessary to loop the ***_anal/detc** or ***_spec/detc** values (for **_pd_instr_dist_**, **_pd_instr_divg_**, **_pd_instr_slit_** and **_pd_instr_soller_**) with the detector IDs (**_pd_calib_detector_id**). It is strongly recommended that the core dictionary term **_diffrn_radiation_probe** (specifying the nature of the radiation used) is employed for all data sets.

Example 1.

```
_pd_instr_slit_eq_src/spec      1.
_pd_instr_slit_eq_anal/detc     0.2

_pd_instr_geometry              Bragg-Brentano
_pd_instr_monochr_post_spec     'graphite (0001)'
_pd_instr_cons_illum_flag       no
```

_pd_instr_beam_size_ax
_pd_instr_beam_size_eq (*numb*)

Axial and equatorial dimensions of the radiation beam at the specimen position (in millimetres). The perpendicular to the plane containing the incident and scattered beam is the axial (***_ax**) direction.

The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_instr_cons_illum_flag (*char*)

Use 'yes' for instruments where the divergence slit is θ-compensated to yield a constant illumination length (also see **_pd_instr_cons_illum_len**). For other flat-plate instruments, where the illumination length changes with 2θ, specify 'no'. Note that if the length is known, it may be specified using **_pd_instr_var_illum_len**.

The data value must be one of the following:

yes
no **[pd_instr]**

_pd_instr_cons_illum_len (*numb*)

Length of the specimen that is illuminated by the radiation source (in millimetres). Use **_pd_instr_cons_illum_len** for instruments where the illumination length does not vary with 2θ, by adjustment of the divergence slits (sometimes known as θ-compensated slits). Use **_pd_instr_var_illum_len** for instruments where the illuminated length of the specimen has been characterized as a function of 2θ, most commonly true with a fixed divergence slit.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_dist_src/mono
_pd_instr_dist_mono/spec
_pd_instr_dist_src/spec
_pd_instr_dist_spec/anal
_pd_instr_dist_anal/detc
_pd_instr_dist_spec/detc (*numb*)

Specifies distances in millimetres for the instrument geometry: ***_src/mono**, the distance from the radiation source to the monochromator; ***_mono/spec**, the distance from the monochromator to the specimen; ***_src/spec**, the distance from the radiation source to the specimen; ***_spec/anal**, the distance from the specimen to the analyser; ***_anal/detc**, the distance from the analyser to the detector; ***_spec/detc**, the distance from the specimen to the detector. Note that ***_src/spec** is used in place of ***_src/mono** and ***_mono/spec** if there is no monochromator in use, and ***_spec/detc** is used in place of ***_spec/anal** and ***_anal/detc** if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_divg_ax_src/mono
_pd_instr_divg_ax_mono/spec
_pd_instr_divg_ax_src/spec
_pd_instr_divg_ax_spec/anal
_pd_instr_divg_ax_anal/detc
_pd_instr_divg_ax_spec/detc (*numb*)

Describes collimation in the axial direction (perpendicular to the plane containing the incident and diffracted beams) for the instrument. Values are the maximum divergence angles in degrees, as limited by slits or beamline optics other than Soller slits (see **_pd_instr_soller_ax_**): ***_src/mono**, collimation between the radiation source and the monochromator; ***_mono/spec**, collimation between the monochromator and the specimen; ***_src/spec**, collimation between the radiation source and the specimen; ***_spec/anal**, collimation between the specimen and the analyser; ***_anal/detc**, collimation between the analyser and the detector; ***_spec/detc**, collimation between the specimen and the detector. Note that ***_src/spec** is used in place of ***_src/mono** and ***_mono/spec** if there is no monochromator in use, and ***_spec/detc** is used in place of ***_spec/anal** and ***_anal/detc** if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_divg_eq_src/mono
_pd_instr_divg_eq_mono/spec
_pd_instr_divg_eq_src/spec
_pd_instr_divg_eq_spec/anal
_pd_instr_divg_eq_anal/detc
_pd_instr_divg_eq_spec/detc (*numb*)

Describes collimation in the equatorial plane (the plane containing the incident and diffracted beams) for the instrument. Values are the maximum divergence angles in degrees, as limited by slits or beamline optics other than Soller slits (see **_pd_instr_soller_eq_**): ***_src/mono**, collimation between the radiation source and the monochromator; ***_mono/spec**, collimation between the monochromator and the specimen; ***_src/spec**, collimation between the radiation source and the specimen; ***_spec/anal**, collimation between the specimen and the analyser; ***_anal/detc**, collimation between the analyser and the detector; ***_spec/detc**, collimation between the specimen and the detector. Note that ***_src/spec** is used in place of ***_src/mono** and ***_mono/spec** if there is no monochromator in use, and

*_spec/detc is used in place of *_spec/anal and *_anal/detc if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_geometry (*char*)

A description of the diffractometer type or geometry.

Examples: 'Bragg-Brentano', 'Guinier',

```
; Parallel-beam non-focusing optics with
  channel-cut monochromator and linear
  position-sensitive detector
;
```

[pd_instr]

_pd_instr_location (*char*)

The name and location of the instrument where measurements were made. This is used primarily to identify data sets measured away from the author's home facility, at shared resources such as a reactor or spallation source.

Example: 'SEPD diffractometer, IPNS, Argonne National Lab (USA)'.

[pd_instr]

_pd_instr_monochr_pre_spec
_pd_instr_monochr_post_spec (*char*)

Indicates the method used to obtain monochromatic radiation. Use **_pd_instr_monochr_pre_spec** to describe the primary beam monochromator (pre-specimen monochromation). Use **_pd_instr_monochr_post_spec** to specify the post-diffraction analyser (post-specimen monochromation). When a monochromator crystal is used, the material and the indices of the Bragg reflection are specified. Note that monochromators may have either 'parallel' or 'antiparallel' orientation. It is assumed that the geometry is parallel unless specified otherwise. In a parallel geometry, the position of the monochromator allows the incident beam and the final post-specimen and post-monochromator beam to be as close to parallel as possible. In a parallel geometry, the diffracting planes in the specimen and monochromator will be parallel when $2\theta_{\text{monochromator}}$ is equal to $2\theta_{\text{specimen}}$. For further discussion see R. Jenkins & R. Snyder (1996). *Introduction to X-ray Powder Diffraction*, pp. 164–165. New York: Wiley.

May appear in list.

Examples: 'Zr filter', 'Ge 220', 'none',
'equatorial mounted graphite (0001)', 'Si (111), antiparallel'.

[pd_instr]

_pd_instr_slit_ax_src/mono
_pd_instr_slit_ax_mono/spec
_pd_instr_slit_ax_src/spec
_pd_instr_slit_ax_spec/anal
_pd_instr_slit_ax_anal/detc
_pd_instr_slit_ax_spec/detc (*numb*)

Describes collimation in the axial direction (perpendicular to the plane containing the incident and diffracted beams) for the instrument as a slit width (as opposed to a divergence angle). Values are the width of the slit (in millimetres) defining: *_src/mono, collimation between the radiation source and the monochromator; *_mono/spec, collimation between the monochromator and the specimen; *_src/spec, collimation between the radiation source and the specimen; *_spec/anal, collimation between the specimen and the analyser; *_anal/detc, collimation between the analyser and the detector; *_spec/detc, collimation between the specimen and the detector. Note that *_src/spec is used in place of *_src/mono and *_mono/spec if there is no monochromator in use, and *_spec/detc is used in place of *_spec/anal and *_anal/detc if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_slit_eq_src/mono
_pd_instr_slit_eq_mono/spec
_pd_instr_slit_eq_src/spec
_pd_instr_slit_eq_spec/anal
_pd_instr_slit_eq_anal/detc
_pd_instr_slit_eq_spec/detc (*numb*)

Describes collimation in the equatorial plane (the plane containing the incident and diffracted beams) for the instrument as a slit width (as opposed to a divergence angle). Values are the width of the slit (in millimetres) defining: *_src/mono, collimation between the radiation source and the monochromator; *_mono/spec, collimation between the monochromator and the specimen; *_src/spec, collimation between the radiation source and the specimen; *_spec/anal, collimation between the specimen and the analyser; *_anal/detc, collimation between the analyser and the detector; *_spec/detc, collimation between the specimen and the detector. Note that *_src/spec is used in place of *_src/mono and *_mono/spec if there is no monochromator in use, and *_spec/detc is used in place of *_spec/anal and *_anal/detc if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_soller_ax_src/mono
_pd_instr_soller_ax_mono/spec
_pd_instr_soller_ax_src/spec
_pd_instr_soller_ax_spec/anal
_pd_instr_soller_ax_anal/detc
_pd_instr_soller_ax_spec/detc (*numb*)

Describes collimation in the axial direction (perpendicular to the plane containing the incident and diffracted beams) for the instrument. Values are the maximum divergence angles in degrees, as limited by Soller slits located thus: *_src/mono, collimation between the radiation source and the monochromator; *_mono/spec, collimation between the monochromator and the specimen; *_src/spec, collimation between the radiation source and the specimen; *_spec/anal, collimation between the specimen and the analyser; *_anal/detc, collimation between the analyser and the detector; *_spec/detc, collimation between the specimen and the detector. Note that *_src/spec is used in place of *_src/mono and *_mono/spec if there is no monochromator in use, and *_spec/detc is used in place of *_spec/anal and *_anal/detc if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_soller_eq_src/mono
_pd_instr_soller_eq_mono/spec
_pd_instr_soller_eq_src/spec
_pd_instr_soller_eq_spec/anal
_pd_instr_soller_eq_anal/detc
_pd_instr_soller_eq_spec/detc (*numb*)

Describes collimation in the equatorial plane (the plane containing the incident and diffracted beams) for the instrument. Values are the maximum divergence angles in degrees, as limited by Soller slits located thus: *_src/mono, collimation between the radiation source and the monochromator; *_mono/spec, collimation between the monochromator and the specimen; *_src/spec, collimation between the radiation source and the specimen; *_spec/anal, collimation between the specimen and the analyser; *_anal/detc, collimation between the analyser and the detector; *_spec/detc, collimation between the specimen and the detector. Note that *_src/spec is used in place of *_src/mono and *_mono/spec if there is no monochromator in use, and *_spec/detc is used in place of *_spec/anal and *_anal/detc if there is no analyser in use.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_instr]**

_pd_instr_source_size_ax
_pd_instr_source_size_eq (*numb*)
Axial and equatorial intrinsic dimensions of the radiation source (in millimetres). The perpendicular to the plane containing the incident and scattered beam is the axial (*_ax) direction.
The permitted range is $0.0 \rightarrow \infty$. [pd_instr]

_pd_instr_special_details (*char*)
A brief description of the instrument giving details that cannot be given in other _pd_instr_ entries.
[pd_instr]

_pd_instr_2theta_monochr_pre
_pd_instr_2theta_monochr_post (*numb*)
The 2θ angle for a pre-specimen or post-specimen monochromator (see _pd_instr_monochr_pre_spec and _pd_instr_monochr_post_spec).
May appear in list.
The permitted range is $-180.0 \rightarrow 180.0$. [pd_instr]

_pd_instr_var_illum_len (*numb*)
Length of the specimen that is illuminated by the radiation source (in millimetres) for instruments where the illumination length varies with 2θ (fixed divergence slits). The _pd_instr_var_illum_len values should be included in the same loop as the intensity measurements (_pd_meas_). See _pd_instr_cons_illum_len for instruments where the divergence slit is θ-compensated to yield a constant illumination length.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. [pd_data]

PD_MEAS

This section contains the measured diffractogram and information about the conditions used for the measurement of the diffraction data set, prior to processing and application of correction terms. While additional information may be added to the CIF as data are processed and transported between laboratories (possibly with the addition of a new _pd_block_id entry), the information in this section of the CIF will rarely be changed once data collection is complete. Where possible, measurements in this section should have no post-collection processing applied (normalizations, corrections, smoothing, zero-offset corrections *etc.*). Such corrected measurements should be recorded in the _pd_proc_ section. Data sets that are measured as counts, where a standard uncertainty can be considered equivalent to the standard deviation and where the standard deviation can be estimated as the square root of the number of counts recorded, should use the _pd_meas_counts_ fields. All other intensity values should be recorded using _pd_meas_intensity_.

Example 1.

```
_pd_meas_info_author_name       'Cranswick, Lachlan'
_pd_meas_info_author_email      lachlan@dmp.csiro.au
_pd_meas_info_author_address    ?
_pd_meas_datetime_initiated     1992-03-23T17:20

_pd_meas_scan_method            step
_pd_meas_2theta_range_min       6.0
_pd_meas_2theta_range_max       164.0
_pd_meas_2theta_range_inc       0.025
_pd_meas_step_count_time        2.0
```

_pd_meas_angle_chi
_pd_meas_angle_omega
_pd_meas_angle_phi
_pd_meas_angle_2theta (*numb*)
The diffractometer angles in degrees for an instrument with an Euler circle. The definitions for these angles follow the convention of *International Tables for X-ray Crystallography* (1974), Vol. IV, p. 276.
May appear in list.
The permitted range is $-180.0 \rightarrow 360.0$. [pd_data]

_pd_meas_counts_total
_pd_meas_counts_background
_pd_meas_counts_container
_pd_meas_counts_monitor (*numb*)
Counts measured at the measurement point as a function of angle, time, channel or some other variable (see _pd_meas_2theta_ *etc.*). The defined fields are: _pd_meas_counts_total, scattering from the specimen (with background, specimen mounting or container scattering included); _pd_meas_counts_background, scattering measured without a specimen, specimen mounting *etc.*, often referred to as the instrument background; _pd_meas_counts_container, the specimen container or mounting without a specimen, includes background; _pd_meas_counts_monitor, counts measured by an incident-beam monitor to calibrate the flux on the specimen. Corrections for background, detector deadtime *etc.* should not have been made to these values. Instead use _pd_proc_intensity_ for corrected diffractograms. Note that counts-per-second values should be converted to total counts. If the counting time varies for different points, it may be included in the loop using _pd_meas_step_count_time. Standard uncertainties should not be quoted for these values. If the standard uncertainties differ from the square root of the number of counts, _pd_meas_intensity_ should be used.
Appears in list.
The permitted range is $0 \rightarrow \infty$. [pd_data]

_pd_meas_datetime_initiated (*char*)
The date and time of the data-set measurement. Entries follow the standard CIF format '*yyyy-mm-dd*T*hh:mm:ss+zz*'. Use of seconds and a time zone is optional, but use of hours and minutes is strongly encouraged. Where possible, give the time when the measurement was started rather than when it was completed.
Example: '1990-07-13T14:40'. [pd_meas_method]

_pd_meas_detector_id (*char*)
A code or number which identifies the measuring detector or channel number in a position-sensitive, energy-dispersive or other multiple-detector instrument. Calibration information, such as angle offsets or a calibration function to convert channel numbers to Q, energy, wavelength, angle *etc.* should be described with _pd_calib_ values. If _pd_calibration_conversion_eqn is used, the detector IDs should be the number to be used in the equation.
Appears in list. **Must** match parent data name _pd_calib_detector_id. [pd_data]

_pd_meas_info_author_address (*char*)
The address of the person who measured the data set. If there is more than one person, this will be looped with _pd_meas_info_author_name.
May appear in list containing _pd_meas_info_author_name. [pd_meas_info]

_pd_meas_info_author_email (*char*)
The e-mail address of the person who measured the data set. If there is more than one person, this will be looped with _pd_meas_info_author_name.
May appear in list containing _pd_meas_info_author_name. [pd_meas_info]

_pd_meas_info_author_fax (*char*)
The fax number of the person who measured the data set. If there is more than one person, this will be looped with **_pd_meas_info_author_name**. The recommended style is the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces.
May appear in list containing **_pd_meas_info_author_name**. **[pd_meas_info]**

_pd_meas_info_author_name (*char*)
The name of the person who measured the data set. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s). For more than one person use a loop to specify multiple values.
May appear in list. **[pd_meas_info]**

_pd_meas_info_author_phone (*char*)
The telephone number of the person who measured the data set. If there is more than one person, this will be looped with **_pd_meas_info_author_name**. The recommended style is the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces.
May appear in list containing **_pd_meas_info_author_name**. **[pd_meas_info]**

_pd_meas_intensity_total
_pd_meas_intensity_background
_pd_meas_intensity_container
_pd_meas_intensity_monitor (*numb, su*)
Intensity measurements at the measurement point (see the definition of **_pd_meas_2theta_**). The defined fields are: **_pd_meas_intensity_total**, scattering from the specimen (with background, specimen mounting or container scattering included); **_pd_meas_intensity_background**, scattering measured without a specimen, specimen mounting *etc.*, often referred to as the instrument background; **_pd_meas_intensity_container**, the specimen container or mounting without a specimen, includes background; **_pd_meas_intensity_monitor**, intensity measured by an incident-beam monitor to calibrate the flux on the specimen. Use these entries for measurements where intensity values are not counts (use **_pd_meas_counts_** for event-counting measurements where the standard uncertainty is estimated as the square root of the number of counts). Corrections for background, detector deadtime *etc.*, should not have been made to these values. Instead use **_pd_proc_intensity_** for corrected diffractograms. **_pd_meas_units_of_intensity** should be used to specify the units of the intensity measurements.
Appears in list. **[pd_data]**

_pd_meas_number_of_points (*numb*)
The total number of points in the measured diffractogram.
The permitted range is $1 \rightarrow \infty$. **[pd_meas_method]**

_pd_meas_point_id (*char*)
Arbitrary label identifying a measured data point. Used to identify a specific entry in a list of measured intensities. The role of this identifier may be adopted by **_pd_data_point_id** if measured, processed and calculated intensity values are combined in a single list.
Related item: **_pd_data_point_id** (alternate). **[pd_data]**

_pd_meas_position (*numb, su*)
A linear distance in millimetres corresponding to the location where an intensity measurement is made. Used for detectors where a distance measurement is made as a direct observable, such as from a microdensitometer trace from film or a strip chart recorder. This is an alternative to **_pd_meas_2theta_scan**, which should only be used for instruments that record intensities directly against 2θ. For instruments where the position scale is nonlinear, the data item **_pd_meas_detector_id** should be used to record positions. Calibration information, such as angle offsets or a function to convert this distance to a 2θ angle or d-space, should be supplied with the **_pd_calib_** values. Do not confuse this with the instrument geometry descriptions given by **_pd_instr_dist_**.
Appears in list. **[pd_data]**

_pd_meas_rocking_angle (*numb*)
The angular range in degrees through which a sample is rotated or oscillated during a measurement step (see **_pd_meas_rocking_axis**).
May appear in list.
The permitted range is $0 \rightarrow 360.0$. **[pd_data]**

_pd_meas_rocking_axis (*char*)
Description of the axis (or axes) used to rotate or rock the specimen for better randomization of crystallites (see **_pd_meas_rocking_angle**).
The data value must be one of the following:
`chi`
`omega`
`phi` **[pd_meas_method]**

_pd_meas_scan_method (*char*)
Code identifying the method for scanning reciprocal space. The designation 'fixed' should be used for measurements where film, a stationary position-sensitive or area detector or other non-moving detection mechanism is used to measure diffraction intensities.
The data value must be one of the following:

`step`	step scan
`cont`	continuous scan
`tof`	time of flight
`disp`	energy dispersive
`fixed`	stationary detector

[pd_meas_method]

_pd_meas_special_details (*char*)
Special details of the diffraction measurement process. Include information about source instability, degradation *etc.* However, this item should not be used to record information that can be specified in other **_pd_meas_** entries.
[pd_meas_method]

_pd_meas_step_count_time (*numb, su*)
The count time in seconds for each intensity measurement. If this value varies for different intensity measurements, then this item will be placed in the loop with the diffraction measurements. If a single fixed value is used, it may be recorded outside the loop.
May appear in list.
The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_meas_2theta_fixed (*numb, su*)
The 2θ diffraction angle in degrees for measurements in a white-beam fixed-angle experiment. For measurements where 2θ is scanned, see **_pd_meas_2theta_scan** or **_pd_meas_2theta_range_**.
The permitted range is $-180.0 \rightarrow 360.0$. **[pd_meas_method]**

_pd_meas_2theta_range_min
_pd_meas_2theta_range_max
_pd_meas_2theta_range_inc (*numb*)
The range of 2θ diffraction angles in degrees for the measurement of intensities. These may be used in place of the **_pd_meas_2theta_scan** values for data sets measured with a constant step size.
The permitted range is $-180.0 \rightarrow 360.0$. **[pd_meas_method]**

_pd_meas_2theta_scan (*numb, su*)
2θ diffraction angle (in degrees) for intensity points measured in a scanning method. The scan method used (*e.g.* continuous or step scan) should be specified in the item **_pd_meas_scan_method**. For fixed 2θ (white-beam) experiments, use **_pd_meas_2theta_fixed**. In the case of continuous-scan data sets, the 2θ value should be the value at the midpoint of the counting period. Associated with each **_pd_meas_2theta_scan** value will be **_pd_meas_counts_** items. The 2θ values should not be corrected for nonlinearity, zero offset *etc.* Corrected values may be specified using **_pd_proc_2theta_corrected**. Note that for data sets collected with constant step size, **_pd_meas_2theta_range_** (***_min**, ***_max** and ***_inc**) may be used instead of **_pd_meas_2theta_scan**.
Appears in list.
The permitted range is $-180.0 \rightarrow 360.0$. **[pd_data]**

_pd_meas_time_of_flight (*numb, su*)
Measured time in microseconds for time-of-flight neutron measurements. Note that the flight distance may be specified using **_pd_instr_dist_** values.
Appears in list.
The permitted range is $0 \rightarrow \infty$. **[pd_data]**

_pd_meas_units_of_intensity (*char*)
Units for intensity measurements when **_pd_meas_intensity_** is used. Note that use of 'counts' or 'counts per second' here is strongly discouraged: convert the intensity measurements to counts and use **_pd_meas_counts_** and **_pd_meas_step_count_time** instead of **_pd_meas_intensity_**.
Examples: 'estimated from strip chart',
'arbitrary, from film density',
'counts, with automatic deadtime correction applied'.
[pd_meas_method]

PD_PEAK

This section contains peak information extracted from the measured or, if present, the processed diffractogram. Each peak in this table will have a unique label (see **_pd_peak_id**). The reflections and phases associated with each peak will be specified in other sections (see the **_pd_refln_** and **_pd_phase_** sections). Note that peak positions are customarily determined from the processed diffractogram and thus corrections for position and intensity will have been previously applied.

_pd_peak_d_spacing (*numb, su*)
Peak position as a *d*-spacing in ångströms.
Appears in list containing **_pd_peak_id**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_peak]**

_pd_peak_id (*char*)
An arbitrary code is assigned to each peak. Used to link with **_pd_refln_peak_id** so that multiple *hkl* and/or phase identifications can be assigned to a single peak. Each peak will have a unique code. In cases where two peaks are severely overlapped, it may be desirable to list them as a single peak. A peak ID must be included for every peak.
Appears in list as essential element of loop structure. May match child data name(s):
_pd_refln_peak_id. **[pd_peak]**

_pd_peak_intensity (*numb, su*)
Integrated area for the peak, with the same scaling as the **_pd_proc_intensity_** values. It is good practice to include s.u.'s for these values.
Appears in list containing **_pd_peak_id**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_peak]**

_pd_peak_pk_height (*numb, su*)
The maximum intensity of the peak, either extrapolated or the highest observed intensity value. The same scaling is used for the **_pd_proc_intensity_** values. It is good practice to include s.u.'s for these values.
Appears in list containing **_pd_peak_id**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_peak]**

_pd_peak_special_details (*char*)
Detailed description of any non-routine processing steps used for peak determination or other comments related to the peak table that cannot be given elsewhere.
[pd_peak_method]

_pd_peak_2theta_centroid
_pd_peak_2theta_maximum (*numb, su*)
Position of the centroid and maximum of a peak as a 2θ angle in degrees.
Appears in list containing **_pd_peak_id**.
The permitted range is $0.0 \rightarrow 180.0$. **[pd_peak]**

_pd_peak_wavelength_id (*char*)
Code identifying the wavelength appropriate for this peak from the wavelengths in the **_diffrn_radiation_** list. (See **_diffrn_radiation_wavelength_id**.) Most commonly used to distinguish $K\alpha_1$ peaks from $K\alpha_2$ or to designate where $K\alpha_1$ and $K\alpha_2$ peaks cannot be resolved. For complex peak tables with multiple superimposed peaks, specify wavelengths in the reflection table using **_pd_refln_wavelength_id** rather than identifying peaks by wavelength.
Appears in list containing **_pd_peak_id**. **Must** match parent data name **_diffrn_radiation_wavelength_id**. **[pd_peak]**

_pd_peak_width_d_spacing (*numb, su*)
Peak width as full-width at half-maximum expressed as a *d*-spacing in ångströms.
Appears in list containing **_pd_peak_id**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_peak]**

_pd_peak_width_2theta (*numb, su*)
Peak width as full-width at half-maximum expressed as a 2θ value in degrees.
Appears in list containing **_pd_peak_id**.
The permitted range is $0.0 \rightarrow 180.0$. **[pd_peak]**

PD_PHASE

This section contains a description of the crystalline phases contributing to the powder diffraction data set. Note that if multiple-phase Rietveld or other structural analysis is performed, the structural results will be placed in different data blocks, using CIF entries from the core CIF dictionary. The **_pd_phase_block_id** entry points to the CIF block with structural parameters for each crystalline phase. The **_pd_phase_id** serves to link to **_pd_refln_phase_id**, which is used to label peaks by phase.

_pd_phase_block_id (*char*)
A block ID code identifying the phase contributing to the diffraction peak. The data block containing the crystallographic information for this phase will be identified with a **_pd_block_id** code matching the code in **_pd_phase_block_id**.
Appears in list containing **_pd_phase_id**. **[pd_phase]**

_pd_phase_id (*char*)
A code for each crystal phase used to link with **_pd_refln_phase_id**.
Appears in list as essential element of loop structure. May match child data name(s): **_pd_refln_phase_id**. **[pd_phase]**

_pd_phase_mass_% (*numb, su*)
Per cent composition of the specified crystal phase expressed as the total mass of the component with respect to the total mass of the specimen.
Appears in list containing **_pd_phase_id**.
The permitted range is $0.0 \rightarrow 100.0$. **[pd_phase]**

_pd_phase_name (*char*)
The name of the crystal phase identified by **_pd_phase_id**. It may be designated as unknown or by a structure type *etc*.
May appear in list containing **_pd_phase_id**. **[pd_phase]**

PD_PREP

This section contains descriptive information about how the sample was prepared.

_pd_prep_conditions (*char*)
A description of how the material was prepared (reaction conditions *etc.*)
[pd_prep]

_pd_prep_cool_rate (*numb, su*)
Cooling rate in kelvins per minute for samples prepared at high temperatures. If the cooling rate is not linear or is unknown (*e.g.* quenched samples), it should be described in **_pd_prep_conditions** instead.
The permitted range is $0.0 \rightarrow \infty$. **[pd_prep]**

_pd_prep_pressure (*numb, su*)
Preparation pressure of the sample in kilopascals. This is particularly important for materials which are metastable at the measurement pressure, **_diffrn_ambient_pressure**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_prep]**

_pd_prep_temperature (*numb, su*)
Preparation temperature of the sample in kelvins. This is particularly important for materials which are metastable at the measurement temperature, **_diffrn_ambient_temperature**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_prep]**

PD_PROC

This section contains the diffraction data set after processing and application of correction terms. If the data set is reprocessed, this section may be replaced (with the addition of a new **_pd_block_id** entry).

_pd_proc_d_spacing (*numb*)
d-spacing corresponding to an intensity point from Bragg's law, $d = \lambda/(2\sin\theta)$, in units of ångströms.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_proc_energy_incident
_pd_proc_energy_detection (*numb*)
Incident energy in electronvolts of the source computed from secondary calibration information (time-of-flight and synchrotron data). Detection energy in electronvolts selected by the analyser, if not the same as the incident energy (triple-axis or energy-dispersive data). This may be a single value or may vary for each data point (triple-axis and time-of-flight data).
May appear in list.
The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_proc_info_author_address (*char*)
The address of the person who processed the data. If there is more than one person, this will be looped with **_pd_proc_info_author_name**.
May appear in list containing **_pd_proc_info_author_name**. **[pd_proc_info]**

_pd_proc_info_author_email (*char*)
The e-mail address of the person who processed the data. If there is more than one person, this will be looped with **_pd_proc_info_author_name**.
May appear in list containing **_pd_proc_info_author_name**. **[pd_proc_info]**

_pd_proc_info_author_fax (*char*)
The fax number of the person who processed the data. If there is more than one person, this will be looped with **_pd_proc_info_author_name**. The recommended style is the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces.
May appear in list containing **_pd_proc_info_author_name**. **[pd_proc_info]**

_pd_proc_info_author_name (*char*)
The name of the person who processed the data, if different from the person(s) who measured the data set. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s). For more than one person use a loop to specify multiple values.
May appear in list. **[pd_proc_info]**

_pd_proc_info_author_phone (*char*)
The telephone number of the person who processed the data. If there is more than one person, this will be looped with **_pd_proc_info_author_name**. The recommended style is the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces.
May appear in list containing **_pd_proc_info_author_name**. **[pd_proc_info]**

_pd_proc_info_data_reduction (*char*)
Description of the processing steps applied in the data-reduction process (background subtraction, α_2 stripping, smoothing *etc.*). Include details of the program(s) used *etc.*
[pd_proc_info]

_pd_proc_info_datetime (*char*)
Date(s) and time(s) when the data set was processed. May be looped if multiple processing steps were used. Dates and times should be specified in the standard CIF format '*yyyy-mm-dd*T*hh*:*mm*:*ss*+*zz*'. Use of seconds and a time zone is optional, but use of hours and minutes is strongly encouraged.
May appear in list.
Example: '1990-07-13T14:40'. **[pd_proc_info]**

_pd_proc_info_excluded_regions (*char*)
Description of regions in the diffractogram excluded from processing along with a justification of why the data points were not used.
Example: '20 to 21 degrees unreliable due to beam dump'.
[pd_proc_info]

_pd_proc_info_special_details (*char*)
Detailed description of any non-routine processing steps applied due to any irregularities in this particular data set.
[pd_proc_info]

_pd_proc_intensity_net
_pd_proc_intensity_total
_pd_proc_intensity_bkg_calc
_pd_proc_intensity_bkg_fix
_pd_proc_intensity_incident
_pd_proc_intensity_norm (*numb, su*)
_pd_proc_intensity_net contains intensity values for the processed diffractogram for each data point (see **_pd_proc_2theta_**, **_pd_proc_wavelength** *etc.*) after correction and normalization factors have been applied (in contrast to **_pd_meas_counts_** values, which are uncorrected). **_pd_proc_intensity_total** contains intensity values for the processed diffractogram for each data point where background, normalization and other corrections have not been applied. Inclusion of s.u.'s for these values is strongly recommended. **_pd_proc_intensity_bkg_calc** is intended to contain the background intensity for every data point where the background function has been fitted or estimated (for example, in all Rietveld and profile fits). If the background is estimated for a limited number of points and the calculated background is then extrapolated from these fixed points, indicate the background values for these points with **_pd_proc_intensity_bkg_fix**. Use a value of '.' for data points where a fixed background has not been defined. The extrapolated background at every point may be specified using **_pd_proc_intensity_bkg_calc**. Background values should be on the same scale as the **_pd_proc_intensity_net** values. Thus normalization and correction factors should be applied before background subtraction (or should be applied to the background values equally). If the intensities have been corrected for a variation of the incident intensity as a function of a data-collection variable (examples: source fluctuations in synchrotrons, θ-compensated slits in conventional diffractometers, spectral corrections for white-beam experiments), the correction function should be specified as **_pd_proc_intensity_incident**. The normalization should be specified in **_pd_proc_intensity_incident** as a value to be used to divide the measured intensities to obtained the normalized diffractogram. Thus, the **_pd_proc_intensity_incident** values should increase as the incident flux is increased. The other normalization factors applied to the data set (for example, Lp corrections, compensation for variation in counting time) may be specified in **_pd_proc_intensity_norm**. The function should be specified as the one used to divide the measured intensities.
Appears in list.
The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_proc_2theta_corrected (*numb*)
The 2θ diffraction angle in degrees of an intensity measurement where 2θ is not constant. Used if corrections such as for nonlinearity, zero offset *etc.* have been applied to the **_pd_meas_2theta_** values or if 2θ values are computed. If the 2θ values are evenly spaced, **_pd_proc_2theta_range_min**, **_pd_proc_2theta_range_max** and **_pd_proc_2theta_range_inc** may be used to specify the 2θ values.
Appears in list.
The permitted range is $-180.0 \rightarrow 180.0$. **[pd_data]**

_pd_proc_2theta_range_min
_pd_proc_2theta_range_max
_pd_proc_2theta_range_inc (*numb*)
The range of 2θ diffraction angles in degrees for the measurement of intensities. These may be used in place of the **_pd_proc_2theta_corrected** values, or in the case of white-beam experiments it will define the fixed 2θ value.
The permitted range is $-180.0 \rightarrow 180.0$. **[pd_proc_info]**

PD_PROC_LS

This section is used to define parameters relevant to a least-squares fit to a powder diffractogram, using a Rietveld or other full-profile (*e.g.* Pawley or Le Bail methods) fit. Note that values in this section refer to full-pattern fitting. Use the appropriate items for single-crystal analyses from the core CIF dictionary for structure refinements using diffraction intensities estimated from a powder diffractogram by pattern-decomposition methods. Also note that many entries in the core **_refine_ls_** entries may also be useful (for example **_refine_ls_shift/su_***).

_pd_proc_ls_background_function (*char*)
Description of the background treatment mechanism used to fit the data set. For refinements where the background is computed as a function that is fitted to minimize the difference between the observed and calculated patterns, it is recommended that in addition to a description of the function (*e.g.* Chebychev polynomial), the actual equation(s) used are included in TEX, or a programming language such as Fortran or C. Include also the values used for the coefficients used in the background function with their s.u.'s. The background values for each data point computed from the function should be specified in **_pd_proc_intensity_bkg_calc**. If background correction is performed using extrapolation from a set of points at fixed locations, these points should be defined using **_pd_proc_intensity_bkg_fix**, and **_pd_proc_ls_background_function** should indicate the extrapolation method (linear extrapolation, spline *etc.*). **_pd_proc_ls_background_function** should also indicate how the points were determined (automatically, by visual estimation *etc.*) and whether the values were refined to improve the agreement. The extrapolated background intensity value for each data point should be specified in **_pd_proc_intensity_bkg_calc**.
[pd_proc_ls]

_pd_proc_ls_peak_cutoff (*numb*)
Describes where peak-intensity computation is discontinued as a fraction of the intensity of the peak at maximum. Thus for a value of 0.005, the tails of a diffraction peak are neglected after the intensity has dropped below 0.5% of the diffraction intensity at the maximum.
[pd_proc_ls]

_pd_proc_ls_pref_orient_corr (*char*)
Description of the preferred-orientation correction if such a correction is used. Omitting this entry implies that no preferred-orientation correction has been used. If a function form is used, it is recommended that the actual equation in TEX, or a programming language, is used to specify the function as well as a giving a description. Include the value(s) used for the correction with s.u.'s.
[pd_proc_ls]

_pd_proc_ls_prof_R_factor
_pd_proc_ls_prof_wR_factor
_pd_proc_ls_prof_wR_expected (*numb*)

Rietveld/profile fit R factors. Note that the R factor computed for Rietveld refinements using the extracted reflection intensity values (often called the Rietveld or Bragg R factor, R_B) is not properly a profile R factor. This R factor may be specified using **_refine_ls_R_I_factor**. (Some authors report **_refine_ls_R_Fsqd_factor** or **_refine_ls_R_factor_all** as the Rietveld or Bragg R factor. While it is appropriate to compute and report any or all of these R factors, the names 'Rietveld or Bragg R factor' refer strictly to **_refine_ls_R_I_factor**.)

_pd_proc_ls_prof_R_factor, often called R_p, is an unweighted fitness metric for the agreement between the observed and computed diffraction patterns.

$$R_p = \sum_i |I_{\rm obs}(i) - I_{\rm calc}(i)| / \sum_i (I_{\rm obs}(i)).$$

_pd_proc_ls_prof_wR_factor, often called $R_{\rm wp}$, is a weighted fitness metric for the agreement between the observed and computed diffraction patterns.

$$R_{\rm wp} = \left(\frac{\sum_i (w(i)[I_{\rm obs}(i) - I_{\rm calc}(i)]^2)}{\sum_i (w(i)[I_{\rm obs}(i)]^2)} \right)^{1/2}.$$

_pd_proc_ls_prof_wR_expected, sometimes called the theoretical $R_{\rm wp}$ or $R_{\rm exp}$, is a weighted fitness metric for the statistical precision of the data set. For an idealized fit, where all deviations between the observed intensities and those computed from the model are due to statistical fluctuations, the observed $R_{\rm wp}$ should match the expected R factor. In reality, $R_{\rm wp}$ will always be higher than $R_{\rm exp}$.

$$R_{\rm exp} = \left(\frac{(n-p)}{\sum_i (w(i)[I_{\rm obs}(i)]^2)} \right)^{1/2}.$$

Note that in the above equations, $w(i)$ is the weight for the ith data point (see **_pd_proc_ls_weight**). $I_{\rm obs}(i)$ is the observed intensity for the ith data point, sometimes referred to as y_i(obs) or y_{oi}. (See **_pd_meas_count_total**, **_pd_meas_intensity_total** or **_pd_proc_intensity_total**.) $I_{\rm calc}(i)$ is the computed intensity for the ith data point with background and other corrections applied to match the scale of the observed data set, sometimes referred to as y_i(calc) or y_{ci}. (See **_pd_calc_intensity_total**.) n is the total number of data points (see **_pd_proc_number_of_points**) less the number of data points excluded from the refinement. p is the total number of refined parameters.

The permitted range is $0.0 \rightarrow \infty$. **[pd_proc_ls]**

_pd_proc_ls_profile_function (*char*)

Description of the profile function used to fit the data set. If a function form is used, it is recommended that the actual equation in TEX, or a programming language, is used to specify the function as well as giving a description. Include the values used for the profile-function coefficients and their s.u.'s.

[pd_proc_ls]

_pd_proc_ls_special_details (*char*)

Additional characterization information relevant to non-routine steps used for refinement of a structural model that cannot be specified elsewhere.

[pd_proc_ls]

_pd_proc_ls_weight (*numb*)

Weight applied to each profile point. These values may be omitted if the weights are $1/u^2$, where u is the s.u. for the **_pd_proc_intensity_net** values. A weight value of zero is used to indicate a data point not used for refinement (see **_pd_proc_info_excluded_regions**).

Appears in list.

The permitted range is $0 \rightarrow \infty$. **[pd_data]**

_pd_proc_number_of_points (*numb*)

The total number of data points in the processed diffractogram.

The permitted range is $1 \rightarrow \infty$. **[pd_proc_info]**

_pd_proc_point_id (*char*)

Arbitrary label identifying a processed data point. Used to identify a specific entry in a list of processed intensities. The role of this identifier may be adopted by **_pd_data_point_id** if measured, processed and calculated intensity values are combined in a single list, or by **_pd_meas_point_id** if measured and processed lists are combined.

Related items:

_pd_data_point_id (alternate),
_pd_meas_point_id (alternate). **[pd_data]**

_pd_proc_recip_len_Q (*numb*)

Length in reciprocal space ($|Q| = 2\pi/d$) corresponding to an intensity point. Units are inverse ångströms.

Appears in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

_pd_proc_wavelength (*numb*)

Wavelength in ångströms for the incident radiation as computed from secondary calibration information. This will be most appropriate for time-of-flight and synchrotron measurements. This will be a single value for continuous-wavelength methods or may vary for each data point and be looped with the intensity values for energy-dispersive measurements.

May appear in list.

The permitted range is $0.0 \rightarrow \infty$. **[pd_data]**

PD_REFLN

This section provides a mechanism to identify each peak in the peak-table section (**_pd_peak_**) with the phase(s) (**_pd_phase_id**) and the reflection indices (**_refln_index_**) associated with the peak. There are no restrictions on the number of phases or reflections associated with an observed peak. Reflections may also be included that are not observed; use '.' for the **_pd_refln_peak_id**.

_pd_refln_peak_id (*char*)

Code which identifies the powder diffraction peak that contains the current reflection. This code must match a **_pd_peak_id** code.

Appears in list containing **_refln_index_h**, **_refln_index_k**, **_refln_index_l**.

Must match parent data name **_pd_peak_id**. **[refln]**

_pd_refln_phase_id (*char*)

Code which identifies the crystal phase associated with this reflection. This code must match a **_pd_phase_id** code.

Appears in list containing **_refln_index_h**, **_refln_index_k**, **_refln_index_l**.

Must match parent data name **_pd_phase_id**. **[refln]**

_pd_refln_wavelength_id (*char*)

Code which identifies the wavelength associated with the reflection and the peak pointed to by **_pd_refln_peak_id**. This code must match a **_diffrn_radiation_wavelength_id** code.

Appears in list containing **_refln_index_h**, **_refln_index_k**, **_refln_index_l**.

Must match parent data name **_diffrn_radiation_wavelength_id**. **[refln]**

PD_SPEC

This section contains information about the specimen used for measurement of the diffraction data set. Note that information about the sample (the batch of material from which the specimen was obtained) is specified in **_pd_prep_**.

Example 1.

```
_pd_spec_mounting            ?
_pd_spec_mount_mode          transmission
_pd_spec_orientation         horizontal
_pd_spec_preparation         ?
```

_pd_spec_description (*char*)
A description of the specimen, such as the source of the specimen, identification of standards, mixtures *etc.*

[pd_spec]

_pd_spec_mount_mode (*char*)
A code describing the beam path through the specimen.
The data value must be one of the following:
- reflection
- transmission

[pd_spec]

_pd_spec_mounting (*char*)
A description of how the specimen is mounted.
Examples: 'vanadium can with He exchange gas', 'quartz capillary', 'packed powder pellet', 'drifted powder on off-cut Si', 'drifted powder on Kapton film'. **[pd_spec]**

_pd_spec_orientation (*char*)
The orientation of the ω (θ) and 2θ axis. Note that this axis is parallel to the specimen axial axis and perpendicular to the plane containing the incident and scattered beams. Thus for a horizontal orientation, scattering measurements are made in a plane perpendicular to the ground (the 2θ axis is parallel to the ground); for vertical orientation, scattering measurements are made in a plane parallel with the ground (the 2θ axis is perpendicular to the ground). 'Both' is appropriate for experiments where measurements are made in both planes, for example using two-dimensional detectors.
The data value must be one of the following:
- horizontal
- vertical
- both

[pd_spec]

_pd_spec_preparation (*char*)
A description of the preparation steps for producing the diffraction specimen from the sample. Include any procedures related to grinding, sieving, spray drying *etc.* For information relevant to how the sample is synthesized, use the **_pd_prep_** entries.
Examples: 'wet grinding in acetone', 'sieved through a 44 micron (325 mesh/inch) sieve', 'spray dried in water with 1% clay'. **[pd_spec]**

_pd_spec_shape (*char*)
A code describing the specimen shape.
The data value must be one of the following:
- cylinder
- flat_sheet
- irregular

[pd_spec]

_pd_spec_size_axial
_pd_spec_size_equat
_pd_spec_size_thick (*numb*)
The size of the specimen in three mutually perpendicular directions in millimetres. The perpendicular to the plane containing the incident and scattered beam is the *_**axial** direction. In transmission geometry, the scattering vector is parallel to *_**equat** and in reflection geometry the scattering vector is parallel to *_**thick**.
The permitted range is $0.0 \rightarrow \infty$. **[pd_spec]**

_pd_spec_special_details (*char*)
Descriptive information about the specimen that cannot be included in other data items.

[pd_spec]

4.3. Modulated and composite structures dictionary (msCIF)

BY G. MADARIAGA

This is version 1.0.1 of the modulated and composite structures CIF dictionary (msCIF). The data names defined in this dictionary complement and/or extend those in the core CIF dictionary (Chapter 4.1) to cover the specific structural features of modulated and composite structures. The msCIF dictionary includes 19 new categories and extends 18 that already exist in the core CIF dictionary. A commentary on the use of this dictionary may be found in Chapter 3.4.

Categories are described in alphabetic order; data items are arranged alphabetically within each category.

ATOM_SITE

Data items in the ATOM_SITE category record details about the atom sites in a crystal structure, such as the positional coordinates, atomic displacement parameters, and magnetic moments and directions. This category exists in the core CIF dictionary but is extended in this dictionary by the addition of some items that may appear in the main looped list of atom-site information.

_atom_site_displace_modulation_flag (*char*)

A code that signals whether the structural model includes the modulation of the positional coordinates of a given atom site.

Appears in list containing **_atom_site_label**.

The data value must be one of the following:

yes	displacive modulation
y	abbreviation for 'yes'
no	no displacive modulation
n	abbreviation for 'no'

Where no value is given, the assumed value is 'no'. **[atom_site]**

_atom_site_occ_modulation_flag (*char*)

A code that signals whether the structural model includes the modulation of the occupation of a given atom site.

Appears in list containing **_atom_site_label**.

The data value must be one of the following:

yes	occupational modulation
y	abbreviation for 'yes'
no	no occupational modulation
n	abbreviation for 'no'

Where no value is given, the assumed value is 'no'. **[atom_site]**

_atom_site_subsystem_code (*char*)

A code that links a given atom or rigid-group site to one of the subsystems present in a composite. This code provides an alternative description for composites which is less explicit than that based on linked data blocks (see the description in this dictionary of AUDIT_LINK). It must match one of the labels specified for **_cell_subsystem_code**.

Appears in list containing **_atom_site_label**. **Must** match parent data name **_cell_subsystem_code**. **[atom_site]**

_atom_site_U_modulation_flag (*char*)

A code that signals whether the structural model includes the modulation of the thermal parameters of a given atom site.

Appears in list containing **_atom_site_label**.

The data value must be one of the following:

yes	modulation of thermal parameters
y	abbreviation for 'yes'
no	no modulation of thermal parameters
n	abbreviation for 'no'

Where no value is given, the assumed value is 'no'. **[atom_site]**

ATOM_SITE_DISPLACE_FOURIER

Data items in the ATOM_SITE_DISPLACE_FOURIER category record details about the Fourier components of the displacive modulation of an atom site in a modulated structure. In the case of rigid groups, items in this category would only include the translational part of the modulation. The rotational part would appear in a separate list of items belonging to the ATOM_SITE_ROT_FOURIER category. The (in general complex) coefficients of each Fourier component belong to the category ATOM_SITE_DISPLACE_FOURIER_PARAM and are listed separately.

*Example 1 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
loop_
    _atom_site_Fourier_wave_vector_seq_id
    _atom_site_Fourier_wave_vector_x
    _atom_site_Fourier_wave_vector_description
         1       0.568          'First harmonic'
         2       1.136          'Second harmonic'
loop_
    _atom_site_displace_Fourier_id
    _atom_site_displace_Fourier_atom_site_label
    _atom_site_displace_Fourier_axis
    _atom_site_displace_Fourier_wave_vector_seq_id
         Nb1z1   Nb1     z       1
         Nb1x2   Nb1     x       2
         Nb1y2   Nb1     y       2
         S1x1    S1      x       1
         S1y1    S1      y       1
         S1z1    S1      z       1
         S1x2    S1      x       2
         S1y2    S1      y       2
         S1z2    S1      z       2
```

*Example 2 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
# NbS2 subsystem has been chosen as reference, i.e. its
# W matrix is the unit matrix.

loop_
    _cell_subsystem_code
    _cell_subsystem_description
    _cell_subsystem_matrix_W_1_1
    _cell_subsystem_matrix_W_1_4
    _cell_subsystem_matrix_W_2_2
    _cell_subsystem_matrix_W_3_3
    _cell_subsystem_matrix_W_4_1
    _cell_subsystem_matrix_W_4_4
         NbS2   '1st subsystem'  1 0 1 1 0 1
         LaS    '2nd subsystem'  0 1 1 1 1 0
```

Affiliation: GOTZON MADARIAGA, Departamento de Física de la Materia Condensada, Facultad de Ciencia y Tecnología, Universidad del País Vasco, Apartado 644, Bilbao 48080, Spain.

```
# The modulation wave vectors are referred to the reciprocal
# basis of each subsystem. They are related to the reciprocal
# basis used to index the whole diffraction pattern through
# the W matrices.

 loop_
     _atom_site_Fourier_wave_vector_seq_id
     _atom_site_Fourier_wave_vector_x
     _atom_site_Fourier_wave_vector_z
     _atom_site_Fourier_wave_vector_description
          1      0.568     0      'First harmonic'
          2      1.136     0      'Second harmonic'
          3      1.761     0.5    'First harmonic'
          4      3.522     1.0    'Second harmonic'

# The modulation coefficients given below are referred to
# each subsystem.

 loop_
     _atom_site_displace_Fourier_id
     _atom_site_displace_Fourier_atom_site_label
     _atom_site_displace_Fourier_axis
     _atom_site_displace_Fourier_wave_vector_seq_id
          Nb1z1_NbS2    Nb1     z    1
          Nb1x2_NbS2    Nb1     x    2
          Nb1y2_NbS2    Nb1     y    2
          S1x1_NbS2     S1      x    1
          S1y1_NbS2     S1      y    1
          S1z1_NbS2     S1      z    1
          S1x2_NbS2     S1      x    2
          S1y2_NbS2     S1      y    2
          S1z2_NbS2     S1      z    2
          La1x3_LaS     La1     x    3
          La1y3_LaS     La1     y    3
          La1z3_LaS     La1     z    3
          La1x4_LaS     La1     x    4
          La1y4_LaS     La1     y    4
          La1z4_LaS     La1     z    4
          S2x3_LaS      S2      x    3
          S2y3_LaS      S2      y    3
          S2z3_LaS      S2      z    3
          S2x4_LaS      S2      x    4
          S2y4_LaS      S2      y    4
          S2z4_LaS      S2      z    4
```

*Example 3 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
# The same structural data but expressed using a set of
# linked data blocks
# Items concerning the modulated structure of the first
# subsystem

data_LaSNbS2_MOD_NbS2

 _audit_block_code     1997-07-24|LaSNbS2|G.M.|_MOD_NbS2

 loop_
     _audit_link_block_code
     _audit_link_block_description
1997-07-24|LaSNbS2|G.M.|
     'common experimental and publication data'
1997-07-24|LaSNbS2|G.M.|_REFRNCE
     'reference structure (common data)'
1997-07-21|LaSNbS2|G.M.|_MOD
     'modulated structure (common data)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_NbS2
     'reference structure (1st subsystem)'
.    'modulated structure (1st subsystem)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_LaS
     'reference structure (2nd subsystem)'
1997-07-21|LaSNbS2|G.M.|_MOD_LaS
     'modulated structure (2nd subsystem)'

 loop_
     _atom_site_Fourier_wave_vector_seq_id
     _atom_site_Fourier_wave_vector_x
     _atom_site_Fourier_wave_vector_description
          1      0.568     'First harmonic'
          2      1.136     'Second harmonic'

 loop_
     _atom_site_displace_Fourier_id
     _atom_site_displace_Fourier_atom_site_label
     _atom_site_displace_Fourier_axis
     _atom_site_displace_Fourier_wave_vector_seq_id
          Nb1z1   Nb1     z       1
          Nb1x2   Nb1     x       2
          Nb1y2   Nb1     y       2
          S1x1    S1      x       1
          S1y1    S1      y       1
          S1z1    S1      z       1
          S1x2    S1      x       2
          S1y2    S1      y       2
          S1z2    S1      z       2
#### End of modulated structure first subsystem data ######

# Items concerning the modulated structure of the second
# subsystem

data_LaSNbS2_MOD_LaS
_audit_block_code          1997-07-24|LaSNbS2|G.M.|_MOD_LaS
 loop_
     _audit_link_block_code
     _audit_link_block_description
1997-07-24|LaSNbS2|G.M.|
     'common experimental and publication data'
1997-07-24|LaSNbS2|G.M.|_REFRNCE
     'reference structure (common data)'
1997-07-21|LaSNbS2|G.M.|_MOD
     'modulated structure (common data)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_NbS2
     'reference structure (1st subsystem)'
1997-07-21|LaSNbS2|G.M.|_MOD_NbS2
     'modulated structure (1st subsystem)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_LaS
     'reference structure (2nd subsystem)'
.    'modulated structure (2nd subsystem)'
 loop_
     _atom_site_Fourier_wave_vector_seq_id
     _atom_site_Fourier_wave_vector_x
     _atom_site_Fourier_wave_vector_z
     _atom_site_Fourier_wave_vector_description
          1      1.761   0.5   'First harmonic'
          2      3.522   1.0   'Second harmonic'
 loop_
     _atom_site_displace_Fourier_id
     _atom_site_displace_Fourier_atom_site_label
     _atom_site_displace_Fourier_axis
     _atom_site_displace_Fourier_wave_vector_seq_id
          La1x1   La1     x       1
          La1y1   La1     y       1
          La1z1   La1     z       1
          La1x2   La1     x       2
          La1y2   La1     y       2
          La1z2   La1     z       2
          S2x1    S2      x       1
          S2y1    S2      y       1
          S2z1    S2      z       1
          S2x2    S2      x       2
          S2y2    S2      y       2
          S2z2    S2      z       2

### End of modulated structure second subsystem data ######
```

*Example 4 – extracted from Baudour & Sanquer [Acta Cryst. (1983), B**39**, 75–84].*

Note the entry from the ATOM_SITES_DISPLACE_FOURIER category to describe collective information relating to all the atom sites.

```
 _atom_sites_displace_Fourier_axes_description
 ;   a1 and a2 are respectively the long molecular axis
     and the axis normal to the mean molecular plane.
 ;

 loop_
     _atom_site_displace_Fourier_id
     _atom_site_displace_Fourier_atom_site_label
     _atom_site_displace_Fourier_axis
     _atom_site_displace_Fourier_wave_vector_seq_id
          Byphenyl_a1_1    Biphenyl  a1    1
```

_atom_site_displace_Fourier_atom_site_label

(*char*)

Modulation parameters are usually looped in separate lists. Modulated parameters are the atom positions (displacive modulation), the atomic occupation (occupational modulation) and/or the atomic anisotropic (or isotropic) displacement parameters (referred to as modulation of thermal parameters, since the term 'displacement parameters' is ambiguous in this context). _atom_site_displace_Fourier_atom_site_label is the code that identifies an atom or rigid group in a loop in which the Fourier components of its displacive modulation are listed. In the case of a rigid group, this list would only include the translational part of its displacive modulation. The rotational part (if any) would appear in a separate list (see _atom_site_rot_Fourier_atom_site_label). This code must match the _atom_site_label of the associated coordinate list and conform to the rules described in _atom_site_label.

Appears in list containing _atom_site_displace_Fourier_id. **Must** match parent data name _atom_site_label. [atom_site_displace_Fourier]

_atom_site_displace_Fourier_axis

(*char*)

A label identifying the displacement component of a given atom or rigid group that is being parameterized by Fourier series. **a**, **b** and **c** are the basic lattice vectors of the reference structure. For composites they refer to the reference structure of each subsystem. $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ are defined by _atom_sites_displace_Fourier_axes_description.

Appears in list containing _atom_site_displace_Fourier_id.

The data value must be one of the following:

x	displacement along the *a* axis
y	displacement along the *b* axis
z	displacement along the *c* axis
a1	displacement along an arbitrary a_1 axis
a2	displacement along an arbitrary a_2 axis
a3	displacement along an arbitrary a_3 axis

[atom_site_displace_Fourier]

_atom_site_displace_Fourier_id

(*char*)

A code identifying each component of the displacive modulation of a given atom or rigid group when the modulation is expressed in terms of Fourier series. In the case of a rigid group, it applies only to the translational part of the distortion.

Appears in list as essential element of loop structure. May match child data name(s): _atom_site_displace_Fourier_param_id.

[atom_site_displace_Fourier]

_atom_site_displace_Fourier_wave_vector_seq_id

(*numb*)

A numeric code identifying the wave vectors of the Fourier terms used in the structural model to describe the displacive modulation of an atom or rigid group. In the case of a rigid group, it applies only to the translational part of the distortion. This code must match _atom_site_Fourier_wave_vector_seq_id.

Appears in list containing _atom_site_displace_Fourier_id. **Must** match parent data name _atom_site_Fourier_wave_vector_seq_id.

[atom_site_displace_Fourier]

ATOM_SITE_DISPLACE_FOURIER_PARAM

Data items in the ATOM_SITE_DISPLACE_FOURIER_PARAM category record details about the coefficients of the Fourier series used to describe the displacive modulation of an atom or rigid group. In the case of rigid groups, items in this category would only include the translational part of the modulation. The rotational part would appear in a separate list of items belonging to the ATOM_SITE_ROT_FOURIER_PARAM category. The Fourier components are defined in the category ATOM_SITE_DISPLACE_FOURIER and are listed separately.

*Example 1 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
loop_
    _atom_site_Fourier_wave_vector_seq_id
    _atom_site_Fourier_wave_vector_x
    _atom_site_Fourier_wave_vector_description
         1       0.568       'First harmonic'
         2       1.136       'Second harmonic'


loop_
    _atom_site_displace_Fourier_id
    _atom_site_displace_Fourier_atom_site_label
    _atom_site_displace_Fourier_axis
    _atom_site_displace_Fourier_wave_vector_seq_id
         Nb1z1   Nb1     z       1
         Nb1x2   Nb1     x       2
         Nb1y2   Nb1     y       2
         S1x1    S1      x       1
         S1y1    S1      y       1
         S1z1    S1      z       1
         S1x2    S1      x       2
         S1y2    S1      y       2
         S1z2    S1      z       2


loop_
    _atom_site_displace_Fourier_param_id
    _atom_site_displace_Fourier_param_cos
    _atom_site_displace_Fourier_param_sin
         Nb1z1   -0.0006(2)   0.
         Nb1x2    0.          0.0078(17)
         Nb1y2   -0.0014(7)   0.
         S1x1     0.         -0.0134(85)
         S1y1    -0.0022(12)  0.
         S1z1     0.0014(14)  0.
         S1x2     0.         -0.0129(27)
         S1y2    -0.0073(27)  0.
         S1z2    -0.0012(3)   0.
```

*Example 2 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
# NbS2 subsystem has been chosen as reference, i.e. its
# W matrix is the unit matrix.


loop_
    _cell_subsystem_code
    _cell_subsystem_description
    _cell_subsystem_matrix_W_1_1
    _cell_subsystem_matrix_W_1_4
    _cell_subsystem_matrix_W_2_2
    _cell_subsystem_matrix_W_3_3
    _cell_subsystem_matrix_W_4_1
    _cell_subsystem_matrix_W_4_4
         NbS2   '1st subsystem'  1 0 1 1 0 1
         LaS    '2nd subsystem'  0 1 1 1 1 0

# The modulation wave vectors are referred to the reciprocal
# basis of each subsystem. They are related to the reciprocal
# basis used to index the whole diffraction pattern through
# the W matrices.
```

```
loop_
     _atom_site_Fourier_wave_vector_seq_id
     _atom_site_Fourier_wave_vector_x
     _atom_site_Fourier_wave_vector_z
     _atom_site_Fourier_wave_vector_description
          1      0.568     0      'First harmonic'
          2      1.136     0      'Second harmonic'
          3      1.761     0.5    'First harmonic'
          4      3.522     1.0    'Second harmonic'

# The modulation coefficients given below are referred to
# each subsystem.

loop_
     _atom_site_displace_Fourier_id
     _atom_site_displace_Fourier_atom_site_label
     _atom_site_displace_Fourier_axis
     _atom_site_displace_Fourier_wave_vector_seq_id
          Nb1z1_NbS2    Nb1     z    1
          Nb1x2_NbS2    Nb1     x    2
          Nb1y2_NbS2    Nb1     y    2
          S1x1_NbS2     S1      x    1
          S1y1_NbS2     S1      y    1
          S1z1_NbS2     S1      z    1
          S1x2_NbS2     S1      x    2
          S1y2_NbS2     S1      y    2
          S1z2_NbS2     S1      z    2
          La1x3_LaS     La1     x    3
          La1y3_LaS     La1     y    3
          La1z3_LaS     La1     z    3
          La1x4_LaS     La1     x    4
          La1y4_LaS     La1     y    4
          La1z4_LaS     La1     z    4
          S2x3_LaS      S2      x    3
          S2y3_LaS      S2      y    3
          S2z3_LaS      S2      z    3
          S2x4_LaS      S2      x    4
          S2y4_LaS      S2      y    4
          S2z4_LaS      S2      z    4

 loop_
     _atom_site_displace_Fourier_param_id
     _atom_site_displace_Fourier_param_cos
     _atom_site_displace_Fourier_param_sin
          Nb1z1_NbS2   -0.0006(2)      0.
          Nb1x2_NbS2    0.             0.0078(17)
          Nb1y2_NbS2   -0.0014(7)      0.
          S1x1_NbS2     0.            -0.0134(85)
          S1y1_NbS2    -0.0022(12)     0.
          S1z1_NbS2     0.0014(14)     0.
          S1x2_NbS2     0.            -0.0129(27)
          S1y2_NbS2    -0.0073(27)     0.
          S1z2_NbS2    -0.0012(3)      0.
          La1x3_LaS     0.            -0.0010(22)
          La1y3_LaS     0.0174(4)      0.
          La1z3_LaS    -0.0005(3)      0.
          La1x4_LaS     0.             0.0144(7)
          La1y4_LaS     0.0001(14)     0.
          La1z4_LaS     0.0008(3)      0.
          S2x3_LaS      0.             0.0059(70)
          S2y3_LaS      0.0081(16)     0.
          S2z3_LaS      0.0009(12)     0.
          S2x4_LaS      0.            -0.0030(30)
          S2y4_LaS      0.0002(56)     0.
          S2z4_LaS      0.0007(10)     0.
```

*Example 3 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
#
# The same structural data but expressed using a set of
# linked data blocks
#

# Items concerning the modulated structure of the first
# subsystem

data_LaSNbS2_MOD_NbS2

 _audit_block_code          1997-07-24|LaSNbS2|G.M.|_MOD_NbS2

 loop_
     _audit_link_block_code
     _audit_link_block_description
1997-07-24|LaSNbS2|G.M.|
     'common experimental and publication data'
1997-07-24|LaSNbS2|G.M.|_REFRNCE
      'reference structure (common data)'
1997-07-21|LaSNbS2|G.M.|_MOD
      'modulated structure (common data)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_NbS2
     'reference structure (1st subsystem)'
.     'modulated structure (1st subsystem)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_LaS
     'reference structure (2nd subsystem)'
1997-07-21|LaSNbS2|G.M.|_MOD_LaS
     'modulated structure (2nd subsystem)'


 loop_
     _atom_site_Fourier_wave_vector_seq_id
     _atom_site_Fourier_wave_vector_x
     _atom_site_Fourier_wave_vector_description
          1      0.568     'First harmonic'
          2      1.136     'Second harmonic'


 loop_
     _atom_site_displace_Fourier_id
     _atom_site_displace_Fourier_atom_site_label
     _atom_site_displace_Fourier_axis
     _atom_site_displace_Fourier_wave_vector_seq_id
          Nb1z1   Nb1     z       1
          Nb1x2   Nb1     x       2
          Nb1y2   Nb1     y       2
          S1x1    S1      x       1
          S1y1    S1      y       1
          S1z1    S1      z       1
          S1x2    S1      x       2
          S1y2    S1      y       2
          S1z2    S1      z       2


 loop_
     _atom_site_displace_Fourier_param_id
     _atom_site_displace_Fourier_param_cos
     _atom_site_displace_Fourier_param_sin
          Nb1z1  -0.0006(2)      0.
          Nb1x2   0.             0.0078(17)
          Nb1y2  -0.0014(7)      0.
          S1x1    0.            -0.0134(85)
          S1y1   -0.0022(12)     0.
          S1z1    0.0014(14)     0.
          S1x2    0.            -0.0129(27)
          S1y2   -0.0073(27)     0.
          S1z2   -0.0012(3)      0.

#### End of modulated structure first subsystem data ######

# Items concerning the modulated structure of the second
# subsystem

data_LaSNbS2_MOD_LaS

 _audit_block_code          1997-07-24|LaSNbS2|G.M.|_MOD_LaS

 loop_
     _audit_link_block_code
     _audit_link_block_description
1997-07-24|LaSNbS2|G.M.|
     'common experimental and publication data'
1997-07-24|LaSNbS2|G.M.|_REFRNCE
     'reference structure (common data)'
1997-07-21|LaSNbS2|G.M.|_MOD
     'modulated structure (common data)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_NbS2
     'reference structure (1st subsystem)'
1997-07-21|LaSNbS2|G.M.|_MOD_NbS2
     'modulated structure (1st subsystem)'
1997-07-24|LaSNbS2|G.M.|_REFRNCE_LaS
     'reference structure (2nd subsystem)'
.    'modulated structure (2nd subsystem)'
```

```
loop_
    _atom_site_Fourier_wave_vector_seq_id
    _atom_site_Fourier_wave_vector_x
    _atom_site_Fourier_wave_vector_z
    _atom_site_Fourier_wave_vector_description
         1      1.761   0.5   'First harmonic'
         2      3.522   1.0   'Second harmonic'

loop_
    _atom_site_displace_Fourier_id
    _atom_site_displace_Fourier_atom_site_label
    _atom_site_displace_Fourier_axis
    _atom_site_displace_Fourier_wave_vector_seq_id
         La1x1   La1     x       1
         La1y1   La1     y       1
         La1z1   La1     z       1
         La1x2   La1     x       2
         La1y2   La1     y       2
         La1z2   La1     z       2
         S2x1    S2      x       1
         S2y1    S2      y       1
         S2z1    S2      z       1
         S2x2    S2      x       2
         S2y2    S2      y       2
         S2z2    S2      z       2

loop_
    _atom_site_displace_Fourier_param_id
    _atom_site_displace_Fourier_param_cos
    _atom_site_displace_Fourier_param_sin
         La1x1   0.             -0.0010(22)
         La1y1   0.0174(4)       0.
         La1z1  -0.0005(3)       0.
         La1x2   0.              0.0144(7)
         La1y2   0.0001(14)      0.
         La1z2   0.0008(3)       0.
         S2x1    0.              0.0059(70)
         S2y1    0.0081(16)      0.
         S2z1    0.0009(12)      0.
         S2x2    0.             -0.0030(30)
         S2y2    0.0002(56)      0.
         S2z2    0.0007(10)      0.

### End of modulated structure second subsystem data ######
```

*Example 4 – extracted from Baudour & Sanquer [Acta Cryst. (1983), B**39**, 75–84].*

Note the entry from the ATOM_SITES_DISPLACE_FOURIER category to describe collective information relating to all the atom sites.

```
_atom_sites_displace_Fourier_axes_description
;   a1 and a2 are respectively the long molecular axis
    and the axis normal to the mean molecular plane.
;

loop_
    _atom_site_displace_Fourier_id
    _atom_site_displace_Fourier_atom_site_label
    _atom_site_displace_Fourier_axis
    _atom_site_displace_Fourier_wave_vector_seq_id
         Byphenyl_a1_1    Biphenyl  a1    1

loop_
    _atom_site_displace_Fourier_param_id
    _atom_site_displace_Fourier_param_modulus
    _atom_site_displace_Fourier_param_phase
         Byphenyl_a1_1    0.035(5)       0.
```

_atom_site_displace_Fourier_param_cos (*numb, su*)
The displacive distortion of a given atom or rigid group (see also **_atom_site_rot_Fourier_param_cos**) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$A_c \cos(2\pi\mathbf{kr}) + A_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|A| \cos(2\pi\mathbf{kr} + \varphi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_displace_Fourier_param_cos** is the cosine coefficient (A_c) corresponding to the Fourier term defined by **_atom_site_displace_Fourier_atom_site_label**, **_atom_site_displace_Fourier_axis** and **_atom_site_displace_Fourier_wave_vector_seq_id**. Atomic or rigid-group displacements must be expressed as fractions of the unit cell or in ångströms if the modulations are referred to some special axes defined by **_atom_sites_displace_Fourier_axes_description**.

Appears in list containing **_atom_site_displace_Fourier_param_id**. Where no value is given, the assumed value is '0.0'.

[atom_site_displace_Fourier_param]

_atom_site_displace_Fourier_param_id (*char*)
A code identifying the (in general complex) coefficient of each term present in the Fourier series describing the displacive modulation of a given atom or rigid group. In the case of a rigid group, it applies only to the translational part of the distortion. This code must match **_atom_site_displace_Fourier_id**.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_displace_Fourier_id**.

[atom_site_displace_Fourier_param]

_atom_site_displace_Fourier_param_modulus (*numb, su*)
The displacive distortion of a given atom or rigid group (see also **_atom_site_rot_Fourier_param_modulus**) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$A_c \cos(2\pi\mathbf{kr}) + A_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|A| \cos(2\pi\mathbf{kr} + \varphi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_displace_Fourier_param_modulus** is the modulus ($|A|$) of the complex amplitude corresponding to the Fourier term defined by **_atom_site_displace_Fourier_atom_site_label**, **_atom_site_displace_Fourier_axis** and **_atom_site_displace_Fourier_wave_vector_seq_id**. Atomic or rigid-group displacements must be expressed as fractions of the unit cell or in ångströms if the modulations are referred to some special axes defined by **_atom_sites_displace_Fourier_axes_description**.

Appears in list containing **_atom_site_displace_Fourier_param_id**.

The permitted range is $0.0 \rightarrow \infty$. Where no value is given, the assumed value is '0.0'.

[atom_site_displace_Fourier_param]

_atom_site_displace_Fourier_param_phase (*numb, su*)
The displacive distortion of a given atom or rigid group (see also **_atom_site_rot_Fourier_param_phase**) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$A_c \cos(2\pi\mathbf{kr}) + A_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|A| \cos(2\pi\mathbf{kr} + \varphi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_displace_Fourier_param_phase** is the phase ($\varphi/2\pi$) in cycles of the complex amplitude

corresponding to the Fourier term defined by **_atom_site_displace_Fourier_atom_site_label**, **_atom_site_displace_Fourier_axis** and **_atom_site_displace_Fourier_wave_vector_seq_id**.

Appears in list containing **_atom_site_displace_Fourier_param_id**.

The permitted range is $-1.0 \rightarrow 1.0$. Where no value is given, the assumed value is '0.0'.

[atom_site_displace_Fourier_param]

_atom_site_displace_Fourier_param_sin (*numb, su*)

The displacive distortion of a given atom or rigid group (see also **_atom_site_rot_Fourier_param_sin**) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$A_c \cos(2\pi\mathbf{kr}) + A_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|A| \cos(2\pi\mathbf{kr} + \varphi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_displace_Fourier_param_sin** is the sine coefficient (A_s) corresponding to the Fourier term defined by **_atom_site_displace_Fourier_atom_site_label**, **_atom_site_displace_Fourier_axis** and **_atom_site_displace_Fourier_wave_vector_seq_id**. Atomic or rigid-group displacements must be expressed as fractions of the unit cell or in ångströms if the modulations are referred to some special axes defined by **_atom_sites_displace_Fourier_axes_description**.

Appears in list containing **_atom_site_displace_Fourier_param_id**. Where no value is given, the assumed value is '0.0'.

[atom_site_displace_Fourier_param]

ATOM_SITE_DISPLACE_SPECIAL_FUNC

Data items in the ATOM_SITE_DISPLACE_SPECIAL_FUNC category record details about the displacive modulation of an atom site in a modulated structure when it is not described by Fourier series. Special functions are effective in some cases where the modulations are highly anharmonic, since the number of parameters is drastically reduced. However, they are in general discontinuous or with discontinuous derivatives and therefore these functions describe an ideal situation that never occurs in a real modulated crystal. Up to now, only a few types of special functions have been used and all of them come from the *JANA* suite of programs. Although this approach is far from being general, it has the advantage that the functions are tightly defined and therefore the atomic displacements and occupations can be calculated easily. In this dictionary, only the special functions available in *JANA*2000 have been included. These are: (1) Sawtooth functions for atomic displacive modulation along *x*, *y* and *z*. (2) Crenel functions for the occupational modulation of atoms and rigid groups. Both of these only apply to one-dimensional modulated structures.

Example 1 – extracted from Gao, Coppens, Cox & Moodenbaugh [Acta Cryst. (1993), A49, 141-148].

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#   In this example the displacive modulation of the O(4)
#   atom was modelled using a sawtooth-shaped function.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 loop_
     _atom_site_displace_special_func_atom_site_label
     _atom_site_displace_special_func_sawtooth_ax
     _atom_site_displace_special_func_sawtooth_ay
     _atom_site_displace_special_func_sawtooth_az
     _atom_site_displace_special_func_sawtooth_c
     _atom_site_displace_special_func_sawtooth_w
      O(4)  -0.270(6)  0.022(9)  0.014(2)  0.42(2)  1.07(2)
```

_atom_site_displace_special_func_atom_site_label (*char*)

The code that identifies an atom in a loop in which the special function that describes its displacive modulation is being defined. This code must match the **_atom_site_label** of the associated coordinate list and conform to the rules described in **_atom_site_label**.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[atom_site_displace_special_func]**

_atom_site_displace_special_func_sawtooth_ax
_atom_site_displace_special_func_sawtooth_ay
_atom_site_displace_special_func_sawtooth_az
_atom_site_displace_special_func_sawtooth_c
_atom_site_displace_special_func_sawtooth_w
(*numb, su*)

_atom_site_displace_special_func_sawtooth_ items are the adjustable parameters of a sawtooth function. A displacive sawtooth function along the internal space is defined as follows:

$$u_x = 2a_x\left(\frac{x_4 - c}{w}\right), \quad u_y = 2a_y\left(\frac{x_4 - c}{w}\right), \quad u_z = 2a_z\left(\frac{x_4 - c}{w}\right)$$

for x_4 belonging to the interval $[c - (w/2), c + (w/2)]$, where a_x, a_y and a_z are the amplitudes (maximum displacements) along each crystallographic axis, w is its width, x_4 is the internal coordinate and c is the centre of the function in internal space. u_x, u_y and u_z must be expressed in relative units. The use of this function is restricted to one-dimensional modulated structures. For more details, see the manual for *JANA*2000 (Petříček & Dušek, 2000).

Reference: Petříček, V. & Dušek, M. (2000). *JANA*2000. *The crystallographic computing system.* Institute of Physics, Prague, Czech Republic.

Appears in list containing **_atom_site_displace_special_func_atom_site_label**. Where no value is given, the assumed value is '0.0'. **[atom_site_displace_special_func]**

ATOM_SITE_FOURIER_WAVE_VECTOR

Data items in the ATOM_SITE_FOURIER_WAVE_VECTOR category record details about the wave vectors of the Fourier terms used in the structural model.

*Example 1 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
 loop_
     _atom_site_Fourier_wave_vector_seq_id
     _atom_site_Fourier_wave_vector_x
     _atom_site_Fourier_wave_vector_description
          1       0.568       'First harmonic'
          2       1.136       'Second harmonic'
```

_atom_site_Fourier_wave_vector_description (*char*)

A description of the linear combination involved in a given Fourier wave vector used to describe the atomic modulation functions.

Appears in list containing **_atom_site_Fourier_wave_vector_seq_id**.

Example: 'q(4)=q(1)+q(2)'. **[atom_site_Fourier_wave_vector]**

_atom_site_Fourier_wave_vector_seq_id (*numb*)
A numeric code identifying the wave vectors defined in **_atom_site_Fourier_wave_vector_**.
Appears in list. May match child data name(s):
_atom_site_displace_Fourier_wave_vector_seq_id,
_atom_site_occ_Fourier_wave_vector_seq_id,
_atom_site_rot_Fourier_wave_vector_seq_id,
_atom_site_U_Fourier_wave_vector_seq_id.
[atom_site_Fourier_wave_vector]

_atom_site_Fourier_wave_vector_x
_atom_site_Fourier_wave_vector_y
_atom_site_Fourier_wave_vector_z (*numb*)
Wave vectors of the Fourier terms used in the structural model to describe the atomic modulation functions, expressed with respect to the three-dimensional reciprocal basis that spans the lattice of main reflections. They are linear combinations with integer coefficients of the independent wave vectors given in the **_cell_wave_vector_** list. Therefore a generic Fourier wave vector is expressed as

$$\mathbf{k} = n(1)\mathbf{q}(1) + \ldots + n(p)\mathbf{q}(p),$$

where p is given by **_cell_modulation_dimension**. In the case of composites described in a single data block, these wave vectors are expressed with respect to the three-dimensional reciprocal basis of each subsystem (see **_cell_subsystem_matrix_W_**).
Appears in list containing **_atom_site_Fourier_wave_vector_seq_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_Fourier_wave_vector]**

ATOM_SITE_OCC_FOURIER

Data items in the ATOM_SITE_OCC_FOURIER category record details about the Fourier components of the occupational modulation of the atom sites in a modulated structure. The (in general complex) coefficients of each Fourier component belong to the category ATOM_SITE_OCC_FOURIER_PARAM and are listed separately.

*Example 1 – extracted from Madariaga, Zúñiga, Paciorek & Bocanegra [Acta Cryst. (1990), B**46**, 620–628].*

```
loop_
    _atom_site_occ_Fourier_id
    _atom_site_occ_Fourier_atom_site_label
    _atom_site_occ_Fourier_wave_vector_seq_id
         CuBr4_1  CuBr4   1
         NC4_1_1  (NC4)1  1
         NC4_2_1  (NC4)2  1
```

_atom_site_occ_Fourier_atom_site_label (*char*)
Modulation parameters are usually looped in separate lists. Modulated parameters are the atom positions (displacive modulation), the atomic occupation (occupational modulation) and/or the atomic anisotropic (or isotropic) displacement parameters (referred to as modulation of thermal parameters, since the term 'displacement parameters' is ambiguous in this context). **_atom_site_occ_Fourier_atom_site_label** is the code that identifies an atom in a loop in which the Fourier components of its occupational modulation are listed. This code must match the **_atom_site_label** of the associated coordinate list and conform to the rules described in **_atom_site_label**.
Appears in list. **Must** match parent data name **_atom_site_label**.
[atom_site_occ_Fourier]

_atom_site_occ_Fourier_id (*char*)
A code identifying each component of the occupational modulation of a given atom or rigid group when the modulation is expressed in terms of Fourier series.
Appears in list as essential element of loop structure. May match child data name(s):
_atom_site_occ_Fourier_param_id. **[atom_site_occ_Fourier]**

_atom_site_occ_Fourier_wave_vector_seq_id (*numb*)
A numeric code identifying the wave vectors of the Fourier terms used in the structural model to describe the modulation functions corresponding to the occupational part of the distortion. This code must match **_atom_site_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_occ_Fourier_id**. **Must** match parent data name **_atom_site_Fourier_wave_vector_seq_id**.
[atom_site_occ_Fourier]

ATOM_SITE_OCC_FOURIER_PARAM

Data items in the ATOM_SITE_OCC_FOURIER_PARAM category record details about the coefficients of the Fourier series used to describe the occupational modulation of the atom sites in a modulated structure. The Fourier components are defined in the category ATOM_SITE_OCC_FOURIER and are listed separately.

*Example 1 – extracted from Madariaga, Zúñiga, Paciorek & Bocanegra [Acta Cryst. (1990), B**46**, 620–628].*

```
loop_
    _atom_site_occ_Fourier_param_id
    _atom_site_occ_Fourier_param_modulus
    _atom_site_occ_Fourier_param_phase
         CuBr4_1  0.397(11)    0.392(6)
         NC4_1_1  0.216(42)   -0.047(33)
         NC4_2_1  0.208(48)    0.132(27)
```

_atom_site_occ_Fourier_param_cos (*numb, su*)
The occupational distortion of a given atom or rigid group is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$P_c \cos(2\pi\mathbf{kr}) + P_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|P| \cos(2\pi\mathbf{kr} + \delta),$$

where $\mathbf{k}$ is the wave vector of the term and $\mathbf{r}$ is the atomic average position. **_atom_site_occ_Fourier_param_cos** is the cosine coefficient (P_c) corresponding to the Fourier term defined by **_atom_site_occ_Fourier_atom_site_label** and **_atom_site_occ_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_occ_Fourier_param_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_occ_Fourier_param]**

_atom_site_occ_Fourier_param_id (*char*)
A code identifying the (in general complex) coefficient of each term present in the Fourier series describing the occupational modulation of a given atom or rigid group. This code must match **_atom_site_occ_Fourier_id**.
Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_occ_Fourier_id**. **[atom_site_occ_Fourier_param]**

_atom_site_occ_Fourier_param_modulus (*numb, su*)
The occupational distortion of a given atom or rigid group is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$P_c \cos(2\pi\mathbf{kr}) + P_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|P| \cos(2\pi\mathbf{kr} + \delta),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_occ_Fourier_param_modulus** is the modulus ($|P|$) of the complex amplitude corresponding to the Fourier term defined by **_atom_site_occ_Fourier_atom_site_label** and **_atom_site_occ_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_occ_Fourier_param_id**.
The permitted range is $0.0 \rightarrow \infty$. Where no value is given, the assumed value is '0.0'.
[atom_site_occ_Fourier_param]

_atom_site_occ_Fourier_param_phase (*numb, su*)
The occupational distortion of a given atom or rigid group is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$P_c \cos(2\pi\mathbf{kr}) + P_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|P| \cos(2\pi\mathbf{kr} + \delta),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_occ_Fourier_param_phase** is the phase ($\delta/2\pi$) in cycles corresponding to the Fourier term defined by **_atom_site_occ_Fourier_atom_site_label** and **_atom_site_occ_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_occ_Fourier_param_id**.
The permitted range is $-1.0 \rightarrow 1.0$. Where no value is given, the assumed value is '0.0'.
[atom_site_occ_Fourier_param]

_atom_site_occ_Fourier_param_sin (*numb, su*)
The occupational distortion of a given atom or rigid group is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$P_c \cos(2\pi\mathbf{kr}) + P_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|P| \cos(2\pi\mathbf{kr} + \delta),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_occ_Fourier_param_sin** is the sine coefficient (P_s) corresponding to the Fourier term defined by **_atom_site_occ_Fourier_atom_site_label** and **_atom_site_occ_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_occ_Fourier_param_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_occ_Fourier_param]**

ATOM_SITE_OCC_SPECIAL_FUNC

Data items in the ATOM_SITE_OCC_SPECIAL_FUNC category record details about the occupational modulation of a given atom or rigid group in a modulated structure when it is not described by Fourier series. Special functions are effective in some cases where the modulations are highly anharmonic, since the number of parameters is drastically reduced. However, they are in general discontinuous or with discontinuous derivatives and therefore these functions describe an ideal situation that never occurs in a real modulated crystal. Up to now, only a few types of special functions have been used and all of them come from the *JANA* suite of programs. Although this approach is far from being general, it has the advantage that the functions are tightly defined and therefore the atomic displacements and occupations can be calculated easily. In this dictionary, only the special functions available in *JANA*2000 have been included. These are: (1) Sawtooth functions for atomic displacive modulation along *x*, *y* and *z*. (2) Crenel functions for the occupational modulation of atoms and rigid groups. Both of these only apply to one-dimensional modulated structures.

Example 1 – extracted from Elding-Pontén, Stenberg, Lidin, Madariaga & Pérez-Mato [Acta Cryst. (1997), B53, 364–372].

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#   In this example the occupational modulation of the Mn(2)
#   atom was modelled using a square-wave crenel function.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 loop_
     _atom_site_occ_special_func_atom_site_label
     _atom_site_occ_special_func_crenel_c
     _atom_site_occ_special_func_crenel_w
         Mn(2)    0.25    0.623(4)
```

_atom_site_occ_special_func_atom_site_label (*char*)
The code that identifies an atom or rigid group in a loop in which the parameters of the special function that describes its occupational modulation are listed. This code must match the **_atom_site_label** of the associated coordinate list and conform to the rules described in **_atom_site_label**.
Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[atom_site_occ_special_func]**

_atom_site_occ_special_func_crenel_c
_atom_site_occ_special_func_crenel_w (*numb, su*)
_atom_site_occ_special_func_crenel_ items are the adjustable parameters of a crenel function. An occupational crenel function along the internal space is defined as follows:

$p(x_4) = 1$ if x_4 belongs to the interval $[c - w/2, c + w/2]$,
$p(x_4) = 0$ if x_4 is outside the interval $[c - w/2, c + w/2]$,

where x_4 is the internal coordinate, c is the centre of the function in internal space and w is its width. The use of this function is restricted to one-dimensional modulated structures. For more details, see the manual for *JANA*2000 (Petříček & Dušek, 2000).

Reference: Petříček, V. & Dušek, M. (2000). *JANA*2000. *The crystallographic computing system.* Institute of Physics, Prague, Czech Republic.

Appears in list containing **_atom_site_occ_special_func_atom_site_label**.
Where no value is given, the assumed value is '0.0'.
[atom_site_occ_special_func]

ATOM_SITE_PHASON

Data items in the ATOM_SITE_PHASON category record details about the atomic phason correction. Although this kind of correction is intended to be overall, some refinement programs (for example, *JANA*2000) allow for this (theoretically dubious) atom-dependent phason treatment.

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
    _atom_site_rot_Fourier_id
    _atom_site_rot_Fourier_atom_site_label
    _atom_site_rot_Fourier_axis
    _atom_site_rot_Fourier_wave_vector_seq_id
         Se04_x_1     Se04    x       1
         Se04_y_1     Se04    y       1
```

*Example 2 – extracted from Baudour & Sanquer [Acta Cryst. (1983), **B39**, 75–84].*

Note the entry from the ATOM_SITES_ROT_FOURIER category to describe collective information relating to all the atom sites.

```
_atom_sites_rot_Fourier_axes_description
;   a1 and a2 are respectively the long molecular axis
    and the axis normal to the mean molecular plane.
;
loop_
    _atom_site_rot_Fourier_id
    _atom_site_rot_Fourier_atom_site_label
    _atom_site_rot_Fourier_axis
    _atom_site_rot_Fourier_wave_vector_seq_id
         Ph1_a1_1  Phenyl1   a1    1
         Ph2_a1_1  Phenyl2   a1    1
         Bph_a2_1  Biphenyl  a2    1
```

_atom_site_phason_atom_site_label (*char*)

The code that identifies an atom or rigid group in a loop in which the phason coefficients are listed. Although this kind of correction is intended to be overall, some refinement programs (for example, *JANA*2000) allow an independent phason correction for each atom or rigid group. In this case, _atom_site_phason_formula and _atom_site_phason_coeff should be used (see also _refine_ls_mod_overall_phason_). This code must match the _atom_site_label of the associated coordinate list and conform to the rules described in _atom_site_label.

Appears in list as essential element of loop structure. **Must** match parent data name _atom_site_label. [atom_site_phason]

_atom_site_phason_coeff (*numb, su*)

The phason coefficient used to calculate (with the appropriate expression given in _atom_site_phason_formula) the atomic phason correction. Although this kind of correction is intended to be overall, some refinement programs (for example, *JANA*2000) allow an independent phason correction for each atom or rigid group. In this case, _atom_site_phason_formula and _atom_site_phason_coeff should be used (see also _refine_ls_mod_overall_phason_).

Appears in list containing _atom_site_phason_atom_site_label.

The permitted range is $0.0 \rightarrow \infty$. Where no value is given, the assumed value is '0.0'.

[atom_site_phason]

_atom_site_phason_formula (*char*)

The formula used for the phason correction. Although both kinds of corrections are intended to be overall, some refinement programs (for example, *JANA*2000) allow an independent phason correction for each atom or rigid group. In this case, _atom_site_phason_formula and _atom_site_phason_coeff should be used (see also _refine_ls_mod_overall_phason_).

Appears in list containing _atom_site_phason_atom_site_label.

The data value must be one of the following:

Axe	Axe, J. D. (1980). *Phys. Rev. B*, **21**, 4181–4190.
Ovr	Overhauser, A. W. (1971). *Phys. Rev. B*, **3**, 3173–3182.

[atom_site_phason]

ATOM_SITE_ROT_FOURIER

Data items in the ATOM_SITE_ROT_FOURIER category record details about the Fourier components present in the rotational part of the displacive modulation of a given rigid group. The translational part would appear in a separate list of items belonging to the ATOM_SITE_DISPLACE_FOURIER category. The (in general complex) coefficients of each Fourier component belong to the category ATOM_SITE_ROT_FOURIER_PARAM and are listed separately.

_atom_site_rot_Fourier_atom_site_label (*char*)

Modulation parameters are usually looped in separate lists. Modulated parameters are the atom positions (displacive modulation), the atomic occupation (occupational modulation) and/or the atomic anisotropic (or isotropic) displacement parameters (referred to as modulation of thermal parameters, since the term 'displacement parameters' is ambiguous in this context). _atom_site_rot_Fourier_atom_site_label is the code that identifies a rigid group in a loop in which the Fourier components of the rotational part of its displacive modulation are listed. The translational part (if any) would appear in a separate list (see _atom_site_displace_Fourier_atom_site_label). This code must match the _atom_site_label of the associated coordinate list and conform to the rules described in _atom_site_label.

Appears in list containing _atom_site_rot_Fourier_id. **Must** match parent data name _atom_site_label. [atom_site_rot_Fourier]

_atom_site_rot_Fourier_axis (*char*)

A label identifying the rotation component around a fixed point of a given rigid group whose modulation is being parameterized by Fourier series. **a**, **b** and **c** are the basic lattice vectors of the reference structure. For composites they refer to the reference structure of each subsystem. $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ are defined by _atom_sites_rot_Fourier_axes_description.

Appears in list containing _atom_site_rot_Fourier_id.

The data value must be one of the following:

x	rotation around the *a* axis
y	rotation around the *b* axis
z	rotation around the *c* axis
a1	rotation around an arbitrary a_1 axis
a2	rotation around an arbitrary a_2 axis
a3	rotation around an arbitrary a_3 axis

[atom_site_rot_Fourier]

_atom_site_rot_Fourier_id (*char*)

A code identifying each component of the rotational modulation of a given rigid group when the modulation is expressed in terms of Fourier series.

Appears in list as essential element of loop structure. May match child data name(s): _atom_site_rot_Fourier_param_id. [atom_site_rot_Fourier]

_atom_site_rot_Fourier_wave_vector_seq_id (*numb*)
A numeric code identifying the wave vectors of the Fourier terms used in the structural model to describe the modulation functions corresponding to the rotational distortion of a rigid group. This code must match **_atom_site_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_rot_Fourier_id**. **Must** match parent data name **_atom_site_Fourier_wave_vector_seq_id**.
[atom_site_rot_Fourier]

_atom_site_rot_Fourier_param_id (*char*)
A code identifying the (in general complex) coefficient of each term present in the Fourier series describing the rotational part of the displacive modulation of a given rigid group. This code must match **_atom_site_rot_Fourier_id**.
Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_rot_Fourier_id**. **[atom_site_rot_Fourier_param]**

ATOM_SITE_ROT_FOURIER_PARAM

Data items in the ATOM_SITE_ROT_FOURIER_PARAM category record details about the coefficients of the Fourier series used to describe the rotational component of the displacive modulation of a given rigid group. The translational part would appear in a separate list of items belonging to the ATOM_SITE_DISPLACE_FOURIER_PARAM category. The Fourier components are defined in the category ATOM_SITE_ROT_FOURIER and are listed separately.

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
    _atom_site_rot_Fourier_param_id
    _atom_site_rot_Fourier_param_cos
    _atom_site_rot_Fourier_param_sin
         SeO4_x_1    -4.2(1)   0.91(3)
         SeO4_y_1     4.3(1)   0.
```

*Example 2 – extracted from Baudour & Sanquer [Acta Cryst. (1983), B**39**, 75–84].*
Note the entry from the ATOM_SITES_ROT_FOURIER category to describe collective information relating to all the atom sites.

```
_atom_sites_rot_Fourier_axes_description
;   a1 and a2 are respectively the long molecular axis
    and the axis normal to the mean molecular plane.
;
loop_
    _atom_site_rot_Fourier_param_id
    _atom_site_rot_Fourier_param_modulus
    _atom_site_rot_Fourier_param_phase
         Ph1_a1_1  11.0(2)  0.
         Ph2_a1_1  11.0(2)  0.5
         Bph_a2_1   1.0(1)  0.25
```

_atom_site_rot_Fourier_param_cos (*numb, su*)
The displacive distortion of a given rigid group is not completely described by **_atom_site_displace_Fourier_**. The rigid rotation of the group around a given axis passing through a fixed point (for example, the centre of mass of the group) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$R_c \cos(2\pi\mathbf{kr}) + R_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|R| \cos(2\pi\mathbf{kr} + \psi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_rot_Fourier_param_cos** is the cosine coefficient (R_c) in degrees corresponding to the Fourier term defined by **_atom_site_rot_Fourier_atom_site_label**, **_atom_site_rot_Fourier_axis** and **_atom_site_rot_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_rot_Fourier_param_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_rot_Fourier_param]**

_atom_site_rot_Fourier_param_modulus (*numb, su*)
The displacive distortion of a given rigid group is not completely described by **_atom_site_displace_Fourier_**. The rigid rotation of the group around a given axis passing through a fixed point (for example, the centre of mass of the group) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$R_c \cos(2\pi\mathbf{kr}) + R_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|R| \cos(2\pi\mathbf{kr} + \psi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_rot_Fourier_param_modulus** is the modulus ($|R|$) in degrees of the complex amplitude corresponding to the Fourier term defined by **_atom_site_rot_Fourier_atom_site_label**, **_atom_site_rot_Fourier_axis** and **_atom_site_rot_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_rot_Fourier_param_id**.
The permitted range is $0.0 \rightarrow \infty$. Where no value is given, the assumed value is '0.0'.
[atom_site_rot_Fourier_param]

_atom_site_rot_Fourier_param_phase (*numb, su*)
The displacive distortion of a given rigid group is not completely described by **_atom_site_displace_Fourier_**. The rigid rotation of the group around a given axis passing through a fixed point (for example, the centre of mass of the group) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$R_c \cos(2\pi\mathbf{kr}) + R_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|R| \cos(2\pi\mathbf{kr} + \psi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_rot_Fourier_param_phase** is the phase ($\psi/2\pi$) in cycles of the complex amplitude corresponding to the Fourier term defined by **_atom_site_rot_Fourier_atom_site_label**, **_atom_site_rot_Fourier_axis** and **_atom_site_rot_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_rot_Fourier_param_id**.
The permitted range is $-1.0 \rightarrow 1.0$. Where no value is given, the assumed value is '0.0'.
[atom_site_rot_Fourier_param]

_atom_site_rot_Fourier_param_sin (*numb, su*)
The displacive distortion of a given rigid group is not completely described by **_atom_site_displace_Fourier_**. The rigid rotation of the group around a given axis passing through a fixed point (for example, the centre of mass of the group) is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$R_c \cos(2\pi\mathbf{kr}) + R_s \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|R| \cos(2\pi\mathbf{kr} + \psi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_rot_Fourier_param_sin** is the sine coefficient (R_s) in degrees corresponding to the Fourier term defined by **_atom_site_rot_Fourier_atom_site_label**, **_atom_site_rot_Fourier_axis** and **_atom_site_rot_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_rot_Fourier_param_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_rot_Fourier_param]**

ATOM_SITE_U_FOURIER

Data items in the ATOM_SITE_U_FOURIER category record details about the Fourier components describing the modulation of the atomic thermal parameters in a modulated structure.

*Example 1 – extracted from Meyer, Paciorek, Schenk, Chapuis & Depmeier [Acta Cryst. (1994), B**50**, 333–343].*

```
loop_
    _atom_site_U_Fourier_id
    _atom_site_U_Fourier_atom_site_label
    _atom_site_U_Fourier_tens_elem
    _atom_site_U_Fourier_wave_vector_seq_id
         Mn_U11_2  Mn  U11  2
         Mn_U22_2  Mn  U22  2
         Mn_U33_2  Mn  U33  2
         Mn_U12_2  Mn  U12  2
         Mn_U13_2  Mn  U13  2
         Mn_U23_2  Mn  U23  2
         Cl1_U11_2 Cl1 U11  2
         Cl1_U22_2 Cl1 U22  2
         Cl1_U33_2 Cl1 U33  2
         Cl1_U12_2 Cl1 U12  2
         Cl1_U13_2 Cl1 U13  2
         Cl1_U23_2 Cl1 U23  2

# - - - - data truncated for brevity - - - -
```

_atom_site_U_Fourier_atom_site_label (*char*)
Modulation parameters are usually looped in separate lists. Modulated parameters are the atom positions (displacive modulation), the atomic occupation (occupational modulation) and/or the atomic anisotropic (or isotropic) displacement parameters (referred to as modulation of thermal parameters, since the term 'displacement parameters' is ambiguous in this context). **_atom_site_U_Fourier_atom_site_label** is the code that identifies an atom in a loop in which the Fourier components of its thermal-parameters modulation are listed. This code must match the **_atom_site_label** of the associated coordinate list and conform to the rules described in **_atom_site_label**.
Appears in list containing **_atom_site_U_Fourier_id**. **Must** match parent data name **_atom_site_label**. **[atom_site_U_Fourier]**

_atom_site_U_Fourier_id (*char*)
A code identifying each Fourier component used to describe the modulation of the atomic thermal parameters.
Appears in list as essential element of loop structure. May match child data name(s): **_atom_site_U_Fourier_param_id**. **[atom_site_U_Fourier]**

_atom_site_U_Fourier_tens_elem (*char*)
A label identifying the temperature tensor element U^{ij} of a given atom or rigid group whose modulation is being parameterized by Fourier series.
Appears in list containing **_atom_site_U_Fourier_id**.
The data value must be one of the following:

U11	modulation of U^{11}
U12	modulation of U^{12}
U13	modulation of U^{13}
U22	modulation of U^{22}
U23	modulation of U^{23}
U33	modulation of U^{33}
Uiso	modulation of $U_{\text{isotropic}}$

[atom_site_U_Fourier]

_atom_site_U_Fourier_wave_vector_seq_id (*numb*)
A numeric code identifying the wave vectors of the Fourier terms used to describe the modulation functions corresponding to the temperature factors of an atom or rigid group. This code must match **_atom_site_Fourier_wave_vector_seq_id**.
Appears in list containing **_atom_site_U_Fourier_id**. **Must** match parent data name **_atom_site_Fourier_wave_vector_seq_id**. **[atom_site_U_Fourier]**

ATOM_SITE_U_FOURIER_PARAM

Data items in the ATOM_SITE_U_FOURIER_PARAM category record details about the coefficients of the Fourier series used to describe the modulation of the atomic thermal parameters in a modulated structure. The Fourier components are defined in the category ATOM_SITE_U_FOURIER and are listed separately.

*Example 1 – extracted from Meyer, Paciorek, Schenk, Chapuis & Depmeier [Acta Cryst. (1994), B**50**, 333–343].*

```
loop_
    _atom_site_U_Fourier_param_id
    _atom_site_U_Fourier_param_modulus
    _atom_site_U_Fourier_param_phase
         Mn_U11_2  0.003(3)  0.0
         Mn_U22_2  0.0       0.0
         Mn_U33_2  0.017(2)  0.0
         Mn_U12_2  0.0       0.0
         Mn_U13_2  0.00(2)   0.5
         Mn_U23_2  0.0       0.0
         Cl1_U11_2 0.003(3)  0.5
         Cl1_U22_2 0.005(3)  0.0
         Cl1_U33_2 0.020(3)  0.0
         Cl1_U12_2 0.008(3)  0.0
         Cl1_U13_2 0.02(2)   0.75
         Cl1_U23_2 0.03(3)   0.25

# - - - - data truncated for brevity - - - -
```

_atom_site_U_Fourier_param_cos (*numb, su*)
The modulation of the atomic thermal parameters is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$U_c^{ij} \cos(2\pi\mathbf{kr}) + U_s^{ij} \sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|U^{ij}| \cos(2\pi\mathbf{kr} + \chi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_U_Fourier_param_cos** is the cosine coefficient U_c^{ij}, in ångströms squared, corresponding to the Fourier term defined by **_atom_site_U_Fourier_atom_site_label**, **_atom_site_U_Fourier_tens_elem** and **_atom_site_U_Fourier_wave_vector_seq_id**.

Appears in list containing **_atom_site_U_Fourier_param_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_U_Fourier_param]**

_atom_site_U_Fourier_param_id (*char*)

A code identifying the (in general complex) coefficient of each term present in the Fourier series describing the modulation of the atomic thermal parameters. This code must match **_atom_site_U_Fourier_id**.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_U_Fourier_id**. **[atom_site_U_Fourier_param]**

_atom_site_U_Fourier_param_modulus (*numb, su*)

The modulation of the atomic thermal parameters is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$U_c^{ij}\cos(2\pi\mathbf{kr}) + U_s^{ij}\sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|U^{ij}|\cos(2\pi\mathbf{kr} + \chi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_U_Fourier_param_modulus** is the modulus $|U^{ij}|$, in ångströms squared, of the complex amplitudes corresponding to the Fourier term defined by **_atom_site_U_Fourier_atom_site_label**, **_atom_site_U_Fourier_tens_elem** and **_atom_site_U_Fourier_wave_vector_seq_id**.

Appears in list containing **_atom_site_U_Fourier_param_id**.

The permitted range is $0.0 \rightarrow \infty$. Where no value is given, the assumed value is '0.0'.

[atom_site_U_Fourier_param]

_atom_site_U_Fourier_param_phase (*numb, su*)

The modulation of the atomic thermal parameters is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$U_c^{ij}\cos(2\pi\mathbf{kr}) + U_s^{ij}\sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|U^{ij}|\cos(2\pi\mathbf{kr} + \chi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_U_Fourier_param_phase** is the phase $(\chi/2\pi)$ in cycles of the complex amplitude corresponding to the Fourier term defined by **_atom_site_U_Fourier_atom_site_label**, **_atom_site_U_Fourier_tens_elem** and **_atom_site_U_Fourier_wave_vector_seq_id**.

Appears in list containing **_atom_site_U_Fourier_param_id**.

The permitted range is $-1.0 \rightarrow 1.0$. Where no value is given, the assumed value is '0.0'.

[atom_site_U_Fourier_param]

_atom_site_U_Fourier_param_sin (*numb, su*)

The modulation of the atomic thermal parameters is usually parameterized by Fourier series. Each term of the series commonly adopts two different representations: the sine–cosine form,

$$U_c^{ij}\cos(2\pi\mathbf{kr}) + U_s^{ij}\sin(2\pi\mathbf{kr}),$$

and the modulus–argument form,

$$|U^{ij}|\cos(2\pi\mathbf{kr} + \chi),$$

where **k** is the wave vector of the term and **r** is the atomic average position. **_atom_site_U_Fourier_param_sin** is the sine coefficient U_s^{ij}, in ångströms squared, corresponding to the Fourier term defined by **_atom_site_U_Fourier_atom_site_label**, **_atom_site_U_Fourier_tens_elem** and **_atom_site_U_Fourier_wave_vector_seq_id**.

Appears in list containing **_atom_site_U_Fourier_param_id**. Where no value is given, the assumed value is '0.0'. **[atom_site_U_Fourier_param]**

ATOM_SITES_DISPLACE_FOURIER

Data items in the ATOM_SITES_DISPLACE_FOURIER category record details common to the displacive modulation of atom sites in a modulated structure. Details for individual atom sites are described by data items in the ATOM_SITE_DISPLACE_FOURIER category.

*Example 1 – extracted from Baudour & Sanquer [Acta Cryst. (1983), B**39**, 75–84].*

```
_atom_sites_displace_Fourier_axes_description
 ;   a1 and a2 are respectively the long molecular axis
     and the axis normal to the mean molecular plane.
 ;
```

_atom_sites_displace_Fourier_axes_description (*char*)

The definition of the axes used for describing the displacive modulation, parameterized by Fourier series, when they are other than the crystallographic axes.

Example:

```
; a1 and a2 are respectively the long molecular axis
  and the axis normal to the mean molecular plane.
;
```

[atom_sites_displace_Fourier]

ATOM_SITES_MODULATION

Data items in the ATOM_SITES_MODULATION category record details common to the modulation of atom sites in a modulated structure.

_atom_sites_modulation_global_phase_t_1
_atom_sites_modulation_global_phase_t_2
_atom_sites_modulation_global_phase_t_3
_atom_sites_modulation_global_phase_t_4
_atom_sites_modulation_global_phase_t_5
_atom_sites_modulation_global_phase_t_6
_atom_sites_modulation_global_phase_t_7
_atom_sites_modulation_global_phase_t_8 (*numb*)

The initial phases, in cycles, of the modulation waves. For incommensurate structures they are irrelevant. However, they are essential for the description of commensurate structures within the superspace formalism, since they determine the space group of the commensurate superstructure [see Perez-Mato, Madariaga, Zuñiga & Garcia Arribas (1987) or van Smaalen (1995)]. Note that for composites described using a single data block, the initial phases for each subsystem are derived using the *W* matrices (see **_cell_subsystem_matrix_W_**) from a unique set of global phases whose values are assigned to **_atom_sites_modulation_global_phase_t_**. Detailed information can be found in van Smaalen (1995).

References: Perez-Mato, J. M., Madariaga, G., Zuñiga, F. J. & Garcia Arribas, A. (1987). *Acta Cryst.* A**43**, 216–226. Smaalen, S. van (1995). *Crystallogr. Rev.* **4**, 79–202.

The permitted range is $-1.0 \rightarrow 1.0$. Where no value is given, the assumed value is '0.0'.

[atom_sites_modulation]

ATOM_SITES_ROT_FOURIER

Data items in the ATOM_SITES_ROT_FOURIER category record details about the rotational component of the displacive modulation of a given rigid group as a whole. Details for individual atom sites are described by data items in the ATOM_SITES_ROT_FOURIER category.

*Example 1 – extracted from Baudour & Sanquer [Acta Cryst. (1983), B**39**, 75–84].*

```
_atom_sites_rot_Fourier_axes_description
;   a1 and a2 are respectively the long molecular axis
    and the axis normal to the mean molecular plane.
;
```

_atom_sites_rot_Fourier_axes_description (*char*)
The definition of the axes used for describing the rotational part of the displacive modulation of a given rigid group, parameterized by Fourier series, when they are other than the crystallographic axes.
Example:

```
; a1 and a2 are respectively the long molecular axis
  and the axis normal to the mean molecular plane.
;                                          [atom_sites_rot_Fourier]
```

AUDIT_LINK

This category description does *not* introduce a new category; instead, it describes the recommended practice for using block codes as described in the core AUDIT_LINK category for descriptions of modulated structures. The value of **_audit_block_code** may be associated with a data block in the same file or in a different file related to the current data block. The value of **_audit_block_code** should be unique.

It is recommended that data blocks are named as follows:

<*string*>: The name of the data block containing those items that, for a particular material, are independent of the specific structure (modulated, reference *etc.*). For example, the experimental set-up or publication details would be described here.

<*string*>_REFRNCE: The name of the data block that contains specific details of the reference (unmodulated) structure if it was refined separately using only main reflections. In the case of composites, this data block may contain those items that are common to the reference structures of all subsystems.

<*string*>_MOD: The name of the data block in which specific details of the modulated structure are given. In the case of composites, this data block may include either those items that are common to the modulated structures of all subsystems or the whole modulated structure if it is described implicitly through the ***_subsystem_code** pointers.

A trailing code following the reserved words MOD or REFRNCE indicates that the corresponding data block includes structural information corresponding to the modulated or reference structures of the subsystem labelled by **_cell_subsystem_code**. A recommended format for <*string*> is given in the definitions of PD_BLOCK and **_pd_block_id** in the dictionary extension cif_pd.dic (Chapter 4.2).

Example 1 – example file for the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
_audit_link_block_code
_audit_link_block_description
.               'publication details'
K2SEO4_COM 'experimental data common to ref./mod. structures'
K2SEO4_REFRNCE  'reference structure'
K2SEO4_MOD      'modulated structure'
```

*Example 2 – example with a trailing string referencing a modulated structure of the subsystem labelled by **_cell_subsystem_code**.*

```
_audit_link_block_code      'PbSVS2_MOD_VS2'
```

CELL

Data items in the CELL category record details about the crystallographic cell parameters and their measurement. This category is already defined in the core CIF dictionary but is extended in this dictionary by the addition of some items that are specific for modulated and composite structures.

_cell_modulation_dimension (*numb*)
Number of additional reciprocal vectors needed to index the whole diffraction pattern using integer Miller indices.
The permitted range is 1 → 8. **[cell]**

_cell_reciprocal_basis_description (*char*)
Definition of the higher-dimensional basis with respect to which the Miller indices are defined. The three-dimensional basis used to index the additional wave vectors should be clearly indicated.
Examples:

```
; a*,b*,c* (reciprocal basis spanning the lattice of main
  reflections) and q (incommensurate with respect to a*,b*,c*)
;  (Typical choice for a one-dimensional incommensurate structure.)
; The diffraction pattern can be indexed with four integers
  based on the reciprocal vectors a*~1~=a*~11~, a*~2~=a*~12~,
  a*~3~=a*~13~, a*~4~=a*~21~. a*~1j~ (j=1,2,3) index the
  main reflections of the 1st subsystem. a*~21~ is
  incommensurate with a*~11~.
;  (Common choice for a misfit layer compound composed of two subsystems that have in
common two reciprocal vectors. Extracted from van Smaalen [Crystallogr. Rev. (1995), 4,
79–202].)                                                  [cell]
```

CELL_SUBSYSTEM

Data items in the CELL_SUBSYSTEM category record details about the crystallographic cell parameters of each subsystem present in a composite.

*Example 1 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
_cell_subsystems_number                  2
loop_
    _cell_subsystem_code
    _cell_subsystem_description
    _cell_subsystem_matrix_W_1_1
    _cell_subsystem_matrix_W_1_4
    _cell_subsystem_matrix_W_2_2
    _cell_subsystem_matrix_W_3_3
    _cell_subsystem_matrix_W_4_1
    _cell_subsystem_matrix_W_4_4
     NbS2    '1st subsystem'   1 0 1 1 0 1
     LaS     '2nd subsystem'   0 1 1 1 1 0
```

_cell_subsystem_code (*char*)
The code identifying uniquely a certain composite subsystem. This code is used to identify the data blocks that contain the structural information associated with the subsystem.
Appears in list.
Example: 'NbS2'. **[cell_subsystem]**

_cell_subsystem_description (*char*)

Description of each subsystem defining a composite structurally. The number of definitions must match the number given in **_cell_subsystems_number**.

Appears in list.

Example: 'NbS2 part of the layer compound (LaS)~1.14~NbS~2~'.

[cell_subsystem]

_cell_subsystem_matrix_W_1_1
_cell_subsystem_matrix_W_1_2
_cell_subsystem_matrix_W_1_3
_cell_subsystem_matrix_W_1_4
_cell_subsystem_matrix_W_1_5
_cell_subsystem_matrix_W_1_6
_cell_subsystem_matrix_W_1_7
_cell_subsystem_matrix_W_1_8
_cell_subsystem_matrix_W_1_9
_cell_subsystem_matrix_W_1_10
_cell_subsystem_matrix_W_1_11
_cell_subsystem_matrix_W_2_1
_cell_subsystem_matrix_W_2_2
_cell_subsystem_matrix_W_2_3
_cell_subsystem_matrix_W_2_4
_cell_subsystem_matrix_W_2_5
_cell_subsystem_matrix_W_2_6
_cell_subsystem_matrix_W_2_7
_cell_subsystem_matrix_W_2_8
_cell_subsystem_matrix_W_2_9
_cell_subsystem_matrix_W_2_10
_cell_subsystem_matrix_W_2_11
_cell_subsystem_matrix_W_3_1
_cell_subsystem_matrix_W_3_2
_cell_subsystem_matrix_W_3_3
_cell_subsystem_matrix_W_3_4
_cell_subsystem_matrix_W_3_5
_cell_subsystem_matrix_W_3_6
_cell_subsystem_matrix_W_3_7
_cell_subsystem_matrix_W_3_8
_cell_subsystem_matrix_W_3_9
_cell_subsystem_matrix_W_3_10
_cell_subsystem_matrix_W_3_11
_cell_subsystem_matrix_W_4_1
_cell_subsystem_matrix_W_4_2
_cell_subsystem_matrix_W_4_3
_cell_subsystem_matrix_W_4_4
_cell_subsystem_matrix_W_4_5
_cell_subsystem_matrix_W_4_6
_cell_subsystem_matrix_W_4_7
_cell_subsystem_matrix_W_4_8
_cell_subsystem_matrix_W_4_9
_cell_subsystem_matrix_W_4_10
_cell_subsystem_matrix_W_4_11
_cell_subsystem_matrix_W_5_1
_cell_subsystem_matrix_W_5_2
_cell_subsystem_matrix_W_5_3
_cell_subsystem_matrix_W_5_4
_cell_subsystem_matrix_W_5_5
_cell_subsystem_matrix_W_5_6
_cell_subsystem_matrix_W_5_7
_cell_subsystem_matrix_W_5_8
_cell_subsystem_matrix_W_5_9
_cell_subsystem_matrix_W_5_10
_cell_subsystem_matrix_W_5_11
_cell_subsystem_matrix_W_6_1
_cell_subsystem_matrix_W_6_2
_cell_subsystem_matrix_W_6_3
_cell_subsystem_matrix_W_6_4
_cell_subsystem_matrix_W_6_5
_cell_subsystem_matrix_W_6_6
_cell_subsystem_matrix_W_6_7
_cell_subsystem_matrix_W_6_8
_cell_subsystem_matrix_W_6_9
_cell_subsystem_matrix_W_6_10
_cell_subsystem_matrix_W_6_11
_cell_subsystem_matrix_W_7_1
_cell_subsystem_matrix_W_7_2
_cell_subsystem_matrix_W_7_3
_cell_subsystem_matrix_W_7_4
_cell_subsystem_matrix_W_7_5
_cell_subsystem_matrix_W_7_6
_cell_subsystem_matrix_W_7_7
_cell_subsystem_matrix_W_7_8
_cell_subsystem_matrix_W_7_9
_cell_subsystem_matrix_W_7_10
_cell_subsystem_matrix_W_7_11
_cell_subsystem_matrix_W_8_1
_cell_subsystem_matrix_W_8_2
_cell_subsystem_matrix_W_8_3
_cell_subsystem_matrix_W_8_4
_cell_subsystem_matrix_W_8_5
_cell_subsystem_matrix_W_8_6
_cell_subsystem_matrix_W_8_7
_cell_subsystem_matrix_W_8_8
_cell_subsystem_matrix_W_8_9
_cell_subsystem_matrix_W_8_10
_cell_subsystem_matrix_W_8_11
_cell_subsystem_matrix_W_9_1
_cell_subsystem_matrix_W_9_2
_cell_subsystem_matrix_W_9_3
_cell_subsystem_matrix_W_9_4
_cell_subsystem_matrix_W_9_5
_cell_subsystem_matrix_W_9_6
_cell_subsystem_matrix_W_9_7
_cell_subsystem_matrix_W_9_8
_cell_subsystem_matrix_W_9_9
_cell_subsystem_matrix_W_9_10
_cell_subsystem_matrix_W_9_11
_cell_subsystem_matrix_W_10_1
_cell_subsystem_matrix_W_10_2
_cell_subsystem_matrix_W_10_3
_cell_subsystem_matrix_W_10_4
_cell_subsystem_matrix_W_10_5
_cell_subsystem_matrix_W_10_6
_cell_subsystem_matrix_W_10_7
_cell_subsystem_matrix_W_10_8
_cell_subsystem_matrix_W_10_9
_cell_subsystem_matrix_W_10_10
_cell_subsystem_matrix_W_10_11
_cell_subsystem_matrix_W_11_1
_cell_subsystem_matrix_W_11_2
_cell_subsystem_matrix_W_11_3
_cell_subsystem_matrix_W_11_4
_cell_subsystem_matrix_W_11_5
_cell_subsystem_matrix_W_11_6
_cell_subsystem_matrix_W_11_7
_cell_subsystem_matrix_W_11_8
_cell_subsystem_matrix_W_11_9
_cell_subsystem_matrix_W_11_10
_cell_subsystem_matrix_W_11_11 (*numb*)

In the case of composites, for each subsystem the matrix W as defined in van Smaalen (1991); see also van Smaalen (1995). Its dimension must match (**_cell_modulation_dimension** + 3) × (**_cell_modulation_dimension** + 3).

Intergrowth compounds are composed of several periodic substructures in which the reciprocal lattices of two different subsystems are incommensurate in at least one direction. The index-

ing of the whole diffraction diagram with integer indices requires more than three reciprocal basic vectors. However, the distinction between main reflections and satellites is not as obvious as in normal incommensurate structures. Indeed, true satellites are normally difficult to locate for composites and the modulation wave vectors are reciprocal vectors of the other subsystem(s) referred to the reciprocal basis of one of them. The choice of the enlarged reciprocal basis $\{\mathbf{a}^*, \mathbf{b}^*, \mathbf{c}^*, \mathbf{q}_1, \ldots, \mathbf{q}_d\}$ is completely arbitrary, but the reciprocal basis of each subsystem is always known through the W matrices. These matrices [$(3+d)\times(3+d)$-dimensional], one for each subsystem, can be blocked as follows:

$$W^{\nu} = \begin{pmatrix} Z_3^{\nu} & Z_d^{\nu} \\ V_3^{\nu} & V_d^{\nu} \end{pmatrix},$$

the dimension of each block being (3×3), $(3 \times d)$, $(d \times 3)$ and $(d \times d)$ for Z_3^{ν}, Z_d^{ν}, V_3^{ν} and V_d^{ν}, respectively. For example, Z^{ν} expresses the reciprocal basis of each subsystem in terms of the basis $\{\mathbf{a}^*, \mathbf{b}^*, \mathbf{c}^*, \mathbf{q}_1, \ldots, \mathbf{q}_d\}$. W^{ν} also gives the irrational components of the modulation wave vectors of each subsystem in its own three-dimensional reciprocal basis $\{\mathbf{a}_{\nu}^*, \mathbf{b}_{\nu}^*, \mathbf{c}_{\nu}^*\}$ and the superspace group of a given subsystem from the unique superspace group of the composite. The structure of these materials is always described by a set of incommensurate structures, one for each subsystem. The atomic coordinates, modulation parameters and wave vectors used for describing the modulation(s) are always referred to the (direct or reciprocal) basis of each particular subsystem. Although expressing the structural results in the chosen common basis is possible (using the matrices W), it is less confusing to use this alternative description. Atomic coordinates are only referred to a common basis when interatomic distances are calculated. Usually, the reciprocal vectors $\{\mathbf{a}^*$, $\mathbf{b}^*$ and $\mathbf{c}^*\}$ span the lattice of main reflections of one of the subsystems and therefore its W matrix is the unit matrix. For composites described in a single data block using *`_subsystem_code` pointers, the cell parameters, the superspace group and the measured modulation wave vectors (see CELL_WAVE_VECTOR below) correspond to the reciprocal basis described in `_cell_reciprocal_basis_description` and coincide with the reciprocal basis of the specific subsystem (if any) whose W matrix is the unit matrix. The cell parameters and the symmetry of the remaining subsystems can be derived using the appropriate W matrices. In any case (single or multiblock CIF), the values assigned to the items describing the atomic parameters (including the wave vectors used to describe the modulations) are always the same and are referred to the basis of each particular subsystem. Such a basis will be explicitly given in a multiblock CIF or should be calculated (with the appropriate W matrix) in the case of a single block description of the composite.

References: Smaalen, S. van (1991). *Phys. Rev. B*, **43**, 11330–11341. Smaalen, S. van (1995). *Crystallogr. Rev.* **4**, 79–202.

Appears in list containing `_cell_subsystem_code`. Where no value is given, the assumed value is '0'. **[cell_subsystem]**

CELL_SUBSYSTEMS

Data items in the CELL_SUBSYSTEMS category describe the gross structure of the subsystems present in a composite.

*Example 1 – based on the modulated structure of inorganic misfit layer $(LaS)_{1.14}NbS_2$ [Smaalen, S. van (1991). J. Phys. Condens. Matter, **3**, 1247–1263].*

```
_cell_subsystems_number                  2
```

`_cell_subsystems_number` (*numb*)

The number of subsystems used to define the structural model of a composite structure.

The permitted range is $2 \rightarrow \infty$. **[cell_subsystems]**

CELL_WAVE_VECTOR

Data items in the CELL_WAVE_VECTOR category list the independent modulation wave vectors $\mathbf{q}_i$. The diffraction vectors are indexed in the form $h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^* + \sum_i (m_i\mathbf{q}_i)$. $\sum_i$ is taken over all wave vectors. In this version of the dictionary, the index i has been restricted to be less than 9.

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
    _cell_wave_vector_seq_id
    _cell_wave_vector_x
           1        0.318(5)
```

`_cell_wave_vector_seq_id` (*numb*)

A numeric code to identify each independent wave vector. These codes define uniquely the reciprocal basis and, therefore, force the order of the Miller indices assigned to intensities, crystal faces *etc.*

Appears in list. **[cell_wave_vector]**

`_cell_wave_vector_x`
`_cell_wave_vector_y`
`_cell_wave_vector_z` (*numb, su*)

Independent modulation wave vector(s) with which the whole diffraction pattern is indexed, expressed as fractions of the three reciprocal basis vectors of the reference structure. In the case of composites, the modulation wave vectors of each subsystem are expressed in terms of the reciprocal basis of its corresponding reference structure. Their number must match `_cell_modulation_dimension`. In the case of composites described in a single data block, the wave vectors are expressed in the three-dimensional basis chosen as reference in `_cell_reciprocal_basis_description`, which would correspond to the subsystem (if any) whose W matrix is the (`_cell_modulation_dimension` + 3) $\times$ (`_cell_modulation_dimension` + 3) unit matrix. In this case, the wave vectors used to describe the modulation of each subsystem are referred to their own reciprocal basis *via* the W matrices (for details see `_cell_subsystem_matrix_W_` and `_atom_site_Fourier_wave_vector_`).

Appears in list containing `_cell_wave_vector_seq_id`. Where no value is given, the assumed value is '0.0'. **[cell_wave_vector]**

CELL_WAVE_VECTORS

Data items in the CELL_WAVE_VECTORS category record details about the set of independent modulation wave vectors $\mathbf{q}_i$ and their measurement. The diffraction vectors are indexed in the form $h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^* + \sum_i (m_i\mathbf{q}_i)$. $\sum_i$ is taken over all wave vectors. In this version of the dictionary, the index i has been restricted to be less than 9.

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
_cell_wave_vectors_meas_details
     'Determined from profiles along q'
```

`_cell_wave_vectors_meas_details` (*char*)

Details about the method used to determine the independent modulation wave vector(s).

[cell_wave_vectors]

_cell_wave_vectors_pressure_max
_cell_wave_vectors_pressure_min (*numb, su*)
The maximum and minimum values of the pressure in kilopascals defining the interval within which the modulation wave vector(s) were measured.
The permitted range is $0.0 \rightarrow \infty$. **[cell_wave_vectors]**

_cell_wave_vectors_temp_max
_cell_wave_vectors_temp_min (*numb, su*)
The maximum and minimum values of the temperature in kelvins defining the interval within which the modulation wave vector(s) were measured.
The permitted range is $0.0 \rightarrow \infty$. **[cell_wave_vectors]**

_cell_wave_vectors_variation (*char*)
Details concerning the behaviour (and its experimental detection) of the wave vector(s) with temperature and/or pressure within the ranges specified by **_cell_wave_vectors_pressure_max**, **_cell_wave_vectors_pressure_min**, **_cell_wave_vectors_temp_max** and **_cell_wave_vectors_temp_min**.
[cell_wave_vectors]

DIFFRN_REFLN

Data items in the DIFFRN_REFLN category record details about the intensities measured in the diffraction experiment. The DIFFRN_REFLN data items refer to individual intensity measurements and must be included in looped lists. (The DIFFRN_REFLNS data items specify the parameters that apply to all intensity measurements. The DIFFRN_REFLNS data items are not looped.) Data items in this category are extensions of the core CIF dictionary definitions to the indexing of diffraction intensities by higher-dimensional components.

_diffrn_refln_index_m_1
_diffrn_refln_index_m_2
_diffrn_refln_index_m_3
_diffrn_refln_index_m_4
_diffrn_refln_index_m_5
_diffrn_refln_index_m_6
_diffrn_refln_index_m_7
_diffrn_refln_index_m_8 (*numb*)
Additional Miller indices needed to write the reciprocal vector of a certain reflection in the basis described in **_cell_reciprocal_basis_description**. Following the usual convention, such a vector would be expressed as

$$\mathbf{H} = h^*\mathbf{a}^* + k^*\mathbf{b}^* + l^*\mathbf{c}^* + m_1^*\mathbf{q}_1 + ... + m_8^*\mathbf{q}_8,$$

where h, k, l are the usual **_diffrn_refln_index_**, and $\mathbf{q}_1, \ldots, \mathbf{q}_8$ represent the independent wave vectors given by **_cell_wave_vector_** and identified by **_cell_wave_vector_seq_id**. Therefore, the total number of indices of a given reflection must match (**_cell_modulation_dimension**+3) and the order of the additional indices must be consistent with the codes given in **_cell_wave_vector_seq_id**. These indices need not match **_refln_index_m_** values if a transformation of the original measured cell has occurred.
Appears in list containing **_diffrn_refln_index_h**, **_diffrn_refln_index_k**, **_diffrn_refln_index_l**. **[diffrn_refln]**

DIFFRN_REFLNS

Data items in the DIFFRN_REFLNS category record details about the set of intensities measured in the diffraction experiment. The DIFFRN_REFLNS data items specify the parameters that apply to all intensity measurements. The DIFFRN_REFLNS data items are not looped. (The DIFFRN_REFLN data items refer to individual intensity measurements and must be included in looped lists.) Data items in this category extend the core CIF dictionary definitions providing independent checks on the range of values recorded for each of the additional Miller indices given in the DIFFRN_REFLN category.

_diffrn_reflns_limit_index_m_1_max
_diffrn_reflns_limit_index_m_1_min
_diffrn_reflns_limit_index_m_2_max
_diffrn_reflns_limit_index_m_2_min
_diffrn_reflns_limit_index_m_3_max
_diffrn_reflns_limit_index_m_3_min
_diffrn_reflns_limit_index_m_4_max
_diffrn_reflns_limit_index_m_4_min
_diffrn_reflns_limit_index_m_5_max
_diffrn_reflns_limit_index_m_5_min
_diffrn_reflns_limit_index_m_6_max
_diffrn_reflns_limit_index_m_6_min
_diffrn_reflns_limit_index_m_7_max
_diffrn_reflns_limit_index_m_7_min
_diffrn_reflns_limit_index_m_8_max
_diffrn_reflns_limit_index_m_8_min (*numb*)
Maximum and minimum values of the additional Miller indices appearing in **_diffrn_refln_index_m_**. The number of ranges must match **_cell_modulation_dimension**. The order of the additional indices must be consistent with the codes given in **_cell_wave_vector_seq_id**.
[diffrn_reflns]

_diffrn_reflns_satellite_order_max (*numb*)
Maximum order of observed satellites.
[diffrn_reflns]

DIFFRN_STANDARD_REFLN

Data items in the DIFFRN_STANDARD_REFLN category record details about the reflections treated as standards during the measurement of diffraction intensities. Note that these are the individual standard reflections, not the results of the analysis of the standard reflections. Data items in this category are extensions of the core CIF dictionary definitions to the indexing of standard reflections by higher-dimensional components.

_diffrn_standard_refln_index_m_1
_diffrn_standard_refln_index_m_2
_diffrn_standard_refln_index_m_3
_diffrn_standard_refln_index_m_4
_diffrn_standard_refln_index_m_5
_diffrn_standard_refln_index_m_6
_diffrn_standard_refln_index_m_7
_diffrn_standard_refln_index_m_8 (*numb*)
Additional Miller indices needed to write the reciprocal vectors of the standard intensities used in the diffraction measurement process, in the basis described in **_cell_reciprocal_basis_description**. The total number of indices of a given standard reflection must match (**_cell_modulation_dimension** + 3) and the

order of the additional indices must be consistent with the codes given in **_cell_wave_vector_seq_id**.

Appears in list containing **_diffrn_standard_refln_index_h**, **_diffrn_standard_refln_index_k**, **_diffrn_standard_refln_index_l**.

[diffrn_standard_refln]

EXPTL_CRYSTAL

Data items in the EXPTL_CRYSTAL category record details about experimental measurements on the crystal or crystals used, such as shape, size and density. The new data item added to this category specifies whether the structure is crystalline, modulated or composite.

_exptl_crystal_type_of_structure (*char*)

The type of structure. This is used to check the consistency of a CIF: the data blocks that are expected and/or certain characteristic parameters depend on whether the material is classified as crystalline (periodic in three dimensions), modulated or composite.

The data value must be one of the following:

`cryst`	crystalline structure
`mod`	modulated structure
`comp`	composite (misfit) structure

Where no value is given, the assumed value is '`cryst`'. **[exptl_crystal]**

EXPTL_CRYSTAL_FACE

Data items in the EXPTL_CRYSTAL_FACE category record details of the crystal faces. Data items in this category are extensions of the core CIF dictionary definitions to the indexing of crystal faces by higher-dimensional components.

_exptl_crystal_face_index_m_1
_exptl_crystal_face_index_m_2
_exptl_crystal_face_index_m_3
_exptl_crystal_face_index_m_4
_exptl_crystal_face_index_m_5
_exptl_crystal_face_index_m_6
_exptl_crystal_face_index_m_7
_exptl_crystal_face_index_m_8 (*numb*)

Additional Miller indices of the crystal face associated with the value **_exptl_crystal_face_perp_dist** when the face is indexed using a multidimensional scheme. The total number of indices must match (**_cell_modulation_dimension** + 3). The order of the indices must be consistent with the codes given in **_cell_wave_vector_seq_id**.

Appears in list containing **_exptl_crystal_face_index_h**, **_exptl_crystal_face_index_k**, **_exptl_crystal_face_index_l**.

[exptl_crystal_face]

GEOM_ANGLE

Data items in the GEOM_ANGLE category record details about the bond angles, as calculated from the ATOM, CELL and SYMMETRY data. These extensions to the core CIF dictionary definitions record the maximum, minimum and average values of angles and extend the symmetry-operation code used in angle listings to the higher-dimensional superspace form.

_geom_angle_max
_geom_angle_min
_geom_angle_av (*numb, su*)

Maximum, minimum and average angles in degrees bounded by **_geom_angle_atom_site_label_1**, *_2, and *_3. The site at *_2 is at the apex of the angle.

Appears in list containing **_geom_angle_atom_site_label_**. **[geom_angle]**

_geom_angle_site_ssg_symmetry_1
_geom_angle_site_ssg_symmetry_2
_geom_angle_site_ssg_symmetry_3 (*char*)

The symmetry code of each atom site as the symmetry operation number 'n' and the higher-dimensional translation '$m_1 \ldots m_p$'. These numbers are combined to form the code '$n\ m_1 \ldots m_p$' or $n\_m_1 \ldots m_p$. The character string $n\_m_1 \ldots m_p$ is composed as follows: 'n' refers to the symmetry operation that is applied to the superspace coordinates. It must match a number given in **_space_group_symop_ssg_id**. '$m_1 \ldots m_p$' refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the angle. These translations $(t_1, \ldots, t_p)$ are related to $(m_1 \ldots m_p)$ by the relations $m_1 = 5 + t_1, \ldots, m_p = 5 + t_p$. By adding 5 to the translations, the use of negative numbers is avoided. The number 'p' must agree with (**_cell_modulation_dimension** + 3). If there are no cell translations, the translation number may be omitted. If no symmetry operations or translations are applicable, then a single full stop '.' is used.

Appears in list containing **_geom_angle_atom_site_label_**.

Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_6455' (7th symmetry position; +**a** on x, -**b** on y). **[geom_angle]**

GEOM_BOND

Data items in the GEOM_BOND category record details about bonds, as calculated from the ATOM, CELL and SYMMETRY data. These extensions to the core CIF dictionary definitions record the maximum, minimum and average lengths of bonds and extend the symmetry-operation code used in bond listings to the higher-dimensional superspace form.

_geom_bond_distance_max
_geom_bond_distance_min
_geom_bond_distance_av (*numb, su*)

Maximum, minimum and average values of the intramolecular bond distance in ångströms.

Appears in list containing **_geom_bond_atom_site_label_**.

The permitted range is $0.0 \rightarrow \infty$. **[geom_bond]**

_geom_bond_site_ssg_symmetry_1
_geom_bond_site_ssg_symmetry_2 (*char*)

The symmetry code of each atom site as the symmetry operation number 'n' and the higher-dimensional translation '$m_1 \ldots m_p$'. These numbers are combined to form the code '$n\ m_1 \ldots m_p$' or $n\_m_1 \ldots m_p$. The character string $n\_m_1 \ldots m_p$ is composed as follows: 'n' refers to the symmetry operation that is applied to the superspace coordinates. It must match a number given in **_space_group_symop_ssg_id**. '$m_1 \ldots m_p$' refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the bond. These translations $(t_1, \ldots, t_p)$ are related to $(m_1 \ldots m_p)$ by the relations $m_1 = 5 + t_1, \ldots, m_p = 5 + t_p$. By adding 5 to the translations, the use of negative numbers is avoided. The number 'p' must agree with (**_cell_modulation_dimension** + 3). If there are no

cell translations, the translation number may be omitted. If no symmetry operations or translations are applicable, then a single full stop '.' is used.

Appears in list containing **_geom_bond_atom_site_label_**.

Examples: '**.**' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_6455' (7th symmetry position; +**a** on *x*, -**b** on *y*). **[geom_bond]**

GEOM_CONTACT

Data items in the GEOM_CONTACT category record details about interatomic contacts, as calculated from the ATOM, CELL and SYMMETRY data. These extensions to the core CIF dictionary definitions record the maximum, minimum and average values of contact distances and extend the symmetry-operation code used in contact-distance listings to the higher-dimensional superspace form.

_geom_contact_distance_max
_geom_contact_distance_min
_geom_contact_distance_av (*numb, su*)

Maximum, minimum and average values of the interatomic contact distance in ångströms.

Appears in list containing **_geom_contact_atom_site_label_**.

The permitted range is $0.0 \rightarrow \infty$. **[geom_contact]**

_geom_contact_site_ssg_symmetry_1
_geom_contact_site_ssg_symmetry_2 (*char*)

The symmetry code of each atom site as the symmetry operation number 'n' and the higher-dimensional translation '$m_1 \ldots m_p$'. These numbers are combined to form the code '$n\ m_1 \ldots m_p$' or $n\_m_1 \ldots m_p$. The character string $n\_m_1 \ldots m_p$ is composed as follows: 'n' refers to the symmetry operation that is applied to the superspace coordinates. It must match a number given in **_space_group_symop_ssg_id**. '$m_1 \ldots m_p$' refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the contact. These translations $(t_1, \ldots, t_p)$ are related to $(m_1 \ldots m_p)$ by the relations $m_1 = 5 + t_1, \ldots, m_p = 5 + t_p$. By adding 5 to the translations, the use of negative numbers is avoided. The number 'p' must agree with (**_cell_modulation_dimension** + 3). If there are no cell translations, the translation number may be omitted. If no symmetry operations or translations are applicable, then a single full stop '.' is used.

Appears in list containing **_geom_contact_atom_site_label_**.

Examples: '**.**' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_6455' (7th symmetry position; +**a** on *x*, -**b** on *y*). **[geom_contact]**

GEOM_TORSION

Data items in the GEOM_TORSION category record details about torsion angles, as calculated from the ATOM, CELL and SYMMETRY data. These extensions to the core CIF dictionary definitions record the maximum, minimum and average values of torsion angles and extend the symmetry-operation code used in torsion-angle listings to the higher-dimensional superspace form.

_geom_torsion_max
_geom_torsion_min
_geom_torsion_av (*numb, su*)

Maximum, minimum and average torsion angles in degrees bounded by the four atom sites identified by the **_geom_torsion_atom_site_label_** codes. These must match labels specified as **_atom_site_label** in the atom list. The torsion-angle definition should be that of Klyne and Prelog.

Reference: Klyne, W. & Prelog, V. (1960). *Experientia*, **16**, 521–523.

Appears in list containing **_geom_torsion_atom_site_label_**. **[geom_torsion]**

_geom_torsion_site_ssg_symmetry_1
_geom_torsion_site_ssg_symmetry_2
_geom_torsion_site_ssg_symmetry_3
_geom_torsion_site_ssg_symmetry_4 (*char*)

The symmetry code of each atom site as the symmetry operation number 'n' and the higher-dimensional translation '$m_1 \ldots m_p$'. These numbers are combined to form the code '$n\ m_1 \ldots m_p$' or $n\_m_1 \ldots m_p$. The character string $n\_m_1 \ldots m_p$ is composed as follows: 'n' refers to the symmetry operation that is applied to the superspace coordinates. It must match a number given in **_space_group_symop_ssg_id**. '$m_1 \ldots m_p$' refer to the translations that are subsequently applied to the symmetry-transformed coordinates to generate the atom used in calculating the angle. These translations $(t_1, \ldots, t_p)$ are related to $(m_1 \ldots m_p)$ by the relations $m_1 = 5 + t_1, \ldots, m_p = 5 + t_p$. By adding 5 to the translations, the use of negative numbers is avoided. The number 'p' must agree with (**_cell_modulation_dimension** + 3). If there are no cell translations, the translation number may be omitted. If no symmetry operations or translations are applicable, then a single full stop '.' is used.

Appears in list containing **_geom_torsion_atom_site_label_**.

Examples: '**.**' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_6455' (7th symmetry position; +**a** on *x*, -**b** on *y*). **[geom_torsion]**

REFINE

Data items in the REFINE category record details about the structure refinement parameters. The new items in this category extend those of the core CIF dictionary and are specific to the refinement of modulated structures.

_refine_ls_mod_func_description (*char*)

Types of modulation present in the structural model and their parameterization.

Examples: 'Only displacive modulation. Fourier series.',

```
; Modulation of atom S(1) described by a non-standard
  linear sawtooth function
;
```

[refine]

_refine_ls_mod_hydrogen_treatment (*char*)

Treatment of hydrogen-atom modulation parameters in the refinement.

The data value must be one of the following:

refA	refined H-atom displacive modulation parameters only
refxyzA	refined H-atom coordinates and displacive modulation parameters only
refP	refined H-atom occupational modulation parameters only
refUP	refined H-atom *U* and occupational modulation parameters only
nomod	no modulation of H-atom parameters

Where no value is given, the assumed value is 'nomod'. **[refine]**

_refine_ls_mod_overall_phason_coeff (*numb, su*)
The phason coefficient used to calculate the overall phason correction.
The permitted range is $0.0 \rightarrow \infty$. Where no value is given, the assumed value is '0.0'.
[refine]

_refine_ls_mod_overall_phason_formula (*char*)
The expression for the overall phason correction, if used.
The data value must be one of the following:

Axe	Axe, J. D. (1980). *Phys. Rev. B*, **21**, 4181–4190.
Ovr	Overhauser, A. W. (1971). *Phys. Rev. B*, **3**, 3173–3182.

[refine]

REFLN

Data items in the REFLN category record details about the reflections used to determine the ATOM_SITE data items. The REFLN data items refer to individual reflections and must be included in looped lists. The REFLNS data items specify the parameters that apply to all reflections. The REFLNS data items are not looped. Data items in this category are extensions of the core CIF dictionary definitions to the indexing of reflections used in the refinement by higher-dimensional components.

_refln_index_m_1
_refln_index_m_2
_refln_index_m_3
_refln_index_m_4
_refln_index_m_5
_refln_index_m_6
_refln_index_m_7
_refln_index_m_8 (*numb*)
Additional Miller indices of a particular reflection in the basis described in **_cell_reciprocal_basis_description**. The total number of indices must match (**_cell_modulation_dimension** + 3). The order of the additional indices must be consistent with the codes given in **_cell_wave_vector_seq_id**.
Appears in list containing **_refln_index_h**, **_refln_index_k**, **_refln_index_l**.
[refln]

REFLNS

Data items in the REFLNS category record details about the reflections used to determine the ATOM_SITE data items. The REFLN data items refer to individual reflections and must be included in looped lists. The REFLNS data items specify the parameters that apply to all reflections. The REFLNS data items are not looped. Data items in this category extend the core CIF dictionary definitions providing independent checks on the range of values recorded for each of the additional Miller indices given in the REFLN category.

_reflns_limit_index_m_1_max
_reflns_limit_index_m_1_min
_reflns_limit_index_m_2_max
_reflns_limit_index_m_2_min
_reflns_limit_index_m_3_max
_reflns_limit_index_m_3_min
_reflns_limit_index_m_4_max
_reflns_limit_index_m_4_min
_reflns_limit_index_m_5_max
_reflns_limit_index_m_5_min
_reflns_limit_index_m_6_max
_reflns_limit_index_m_6_min
_reflns_limit_index_m_7_max
_reflns_limit_index_m_7_min
_reflns_limit_index_m_8_max
_reflns_limit_index_m_8_min (*numb*)
Maximum and minimum values of the additional Miller indices appearing in **_refln_index_m_**. The number of ranges must match **_cell_modulation_dimension**. The order of the additional indices must be consistent with the codes given in **_cell_wave_vector_seq_id**. These need not be the same as the **_diffrn_reflns_limit_index_m_**.
[reflns]

SPACE_GROUP

The SPACE_GROUP category introduced in the symmetry CIF dictionary (cif_sym.dic) is intended to replace the original core SYMMETRY category. For modulated structures, superspace-group descriptions may be included in the same category, but include the **_ssg_** flag to indicate their dimensionality of > 3.

_space_group_ssg_IT_number (*numb*)
Superspace-group number from *International Tables for Crystallography*, Vol. C (2004). Valid only for one-dimensional modulated structures.
Reference: *International Tables for Crystallography* (2004). Vol. C, Chapter 9.8. Dordrecht: Kluwer Academic Publishers.
The permitted range is $1.1 \rightarrow \infty$. **[space_group]**

_space_group_ssg_name (*char*)
Superspace-group symbol conforming to an alternative definition from that given in **_space_group_ssg_name_IT** and **_space_group_ssg_name_WJJ** for one-dimensional modulated structures or to the superspace-group name for higher dimensions. When necessary, indicate the origin and the setting. Use a colon ':' as a separator between the different parts of the superspace-group symbol. Within each part, leave a space between each component. Rules for the notation for Hermann–Mauguin and Hall symbols (if present) are given in the symmetry CIF dictionary (cif_sym.dic) and, partially, in **_space_group_ssg_name_IT** and **_space_group_ssg_name_WJJ**. For composites described in a single data block, the superspace group describes the symmetry of the whole structure. The symmetry of each subsystem can be derived using the appropriate *W* matrices.
Example: 'Hall's notation W:-P -2xb -2ya:q q'. **[space_group]**

_space_group_ssg_name_IT (*char*)
Superspace-group symbol as given in *International Tables for Crystallography*, Vol. C (2004). Valid only for one-dimensional modulated structures. The symbol is divided into three parts: the Hermann–Mauguin space-group symbol of the reference structure, the modulation wave vector and the phase shift (or internal translation) associated with each component of the space group. Each component of the space-group name is separated by a space. Subscripts should appear without special symbols and bars should be given as negative signs. The components of the modulation wave vector (in parentheses) and the phase shifts are also separated by a

space. For composites described in a single data block, the superspace group describes the symmetry of the whole structure. The symmetry of each subsystem can be derived using the appropriate *W* matrices.

Reference: *International Tables for Crystallography* (2004). Vol. C, Chapter 9.8. Dordrecht: Kluwer Academic Publishers.

Example: '`P n m a (0 0 \g) 0 s 0`'. **[space_group]**

_space_group_ssg_name_WJJ (*char*)

Superspace-group symbol as given by de Wolff, Janssen & Janner (1981). Valid only for one-dimensional modulated structures. The symbol is divided into three parts separated by colons ':': the superspace lattice symbol, the Hermann–Mauguin space-group symbol of the reference structure and the phase shift (or internal translation) associated with each component of the space group. Each component of the space-group name is separated by a space. Subscripts should appear without special symbols and bars should be given as negative signs. The phase shifts are also separated by a space. For composites described in a single data block, the superspace group describes the symmetry of the whole structure. The symmetry of each subsystem can be derived using the appropriate *W* matrices.

Reference: Wolff, P. M. de, Janssen, T. & Janner, A. (1981). *Acta Cryst.* A**37**, 625–636.

Example: '`P:P c m n:s s -1`'. **[space_group]**

_space_group_ssg_WJJ_code (*char*)

Superspace-group code as given by de Wolff, Janssen & Janner (1981). Valid only for one-dimensional modulated structures.

Reference: Wolff, P. M. de, Janssen, T. & Janner, A. (1981). *Acta Cryst.* A**37**, 625–636.

Example: '`28a.10.1/2`'. **[space_group]**

SPACE_GROUP_SYMOP

The SPACE_GROUP_SYMOP category introduced in the symmetry CIF dictionary (cif_sym.dic) is intended to replace the original core SYMMETRY_EQUIV category. It contains information about the symmetry operations of the space group. For modulated structures, superspace-group descriptions may be included in the same category, but include the **_ssg_** flag to indicate their dimensionality of > 3.

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
     _space_group_symop_ssg_id
     _space_group_symop_ssg_operation_algebraic
            1          x1,x2,x3,x4
            2          1/2+x1,1/2-x2,1/2-x3,x4
            3          1/2-x1,1/2+x2,-x3,1/2-x4
            4          -x1,-x2,1/2+x3,1/2-x4
            5          -x1,-x2,-x3,-x4
            6          1/2-x1,1/2+x2,1/2+x3,-x4
            7          1/2+x1,1/2-x2,x3,1/2+x4
            8          x1,x2,1/2-x3,1/2+x4
```

_space_group_symop_ssg_id (*numb*)

A numeric code identifying each entry in the **_space_group_symop_ssg_operation_algebraic** list.

Appears in list. **[space_group_symop]**

_space_group_symop_ssg_operation_algebraic (*char*)

A parsable string giving one of the symmetry operations of the superspace group in algebraic form. These data will generally be repeated in a loop. Use symbols as necessary according to **_cell_modulation_dimension**. All symmetry operations should be entered, including the identity operation, those for lattice centring and a centre of symmetry, if present. The symbolic notation for coordinates is such that the identity operation is expressed as $x_1, x_2, x_3, \ldots, x_n$. **_space_group_symop_ssg_operation_algebraic** must always be present in a CIF corresponding to a modulated structure.

Appears in list containing **_space_group_symop_ssg_id**.

Example: '`x1,-x2,x3,1/2+x4`'. **[space_group_symop]**

4.4. Electron density dictionary (rhoCIF)

BY P. R. MALLINSON

This is version 1.0.1 of the electron density CIF dictionary (rhoCIF). The data names defined in this dictionary complement those in the core CIF dictionary (Chapter 4.1).

Categories are described in alphabetical order; data items are arranged alphabetically within each category.

ATOM_LOCAL_AXES

This category allows the definition of local axes around each atom in terms of vectors between neighbouring atoms. High-resolution X-ray diffraction methods enable the determination of the electron density distribution in crystal lattices and molecules, which in turn allows for a characterization of chemical interactions (Coppens, 1997; Koritsanszky & Coppens, 2001). This is accomplished by the construction of a mathematical model of the charge density in a crystal and then by fitting the parameters of such a model to the experimental pattern of diffracted X-rays. The model on which this dictionary is based is the so-called multipole formalism proposed by Hansen & Coppens (1978). In this model, the electron density in a crystal is described by a sum of aspherical 'pseudoatoms' where the pseudoatom density has the form defined in the `_atom_rho_multipole_*` items. Each pseudoatom density consists of terms representing the core density, the spherical part of the valence density and the deviation of the valence density from sphericity. The continuous electron density in the crystal is then modelled as a sum of atom-centred charge distributions. Once the experimental electron density has been established, the 'atoms in molecules' theory of Bader (1990) provides tools for the interpretation of the density distribution in terms of its topological properties.

References: Bader, R. F. W. (1990). *Atoms in molecules: a quantum theory*. Oxford University Press. Coppens, P. (1997). *X-ray charge densities and chemical bonding*. Oxford University Press. Hansen, N. K. & Coppens, P. (1978). *Acta Cryst*. A**34**, 909–921. Koritsanszky, T. S. & Coppens, P. (2001). *Chem. Rev*. **101**, 1583–1621.

Example 1 – This example shows how the local axes can be defined around each atom in terms of vectors between neighbouring atoms.
If necessary, dummy atoms can be introduced into the atom_site list for this purpose.

```
loop_
_atom_local_axes_atom_label
_atom_local_axes_atom0
_atom_local_axes_ax1
_atom_local_axes_atom1
_atom_local_axes_atom2
_atom_local_axes_ax2
    Ni2+(1)  DUM0      Z    Ni2+(1)  N(1)      X

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_occupancy
    DUM0     0.80000     0.80000     0.80000    0.0
```

`_atom_local_axes_atom_label` (*char*)
This item is used to identify an atom for which a local axis system is to be defined. Its value must be identical to one of the values given in the `_atom_site_label` list.
Appears in list as essential element of loop structure. **Must** match parent data name `_atom_site_label`. `[atom_local_axes]`

`_atom_local_axes_atom0` (*char*)
Specifies 'atom0' in the definition of a local axis frame. The definition employs three atom-site labels, 'atom0', 'atom1' and 'atom2', and two axis labels, 'ax1' and 'ax2', having values '$\pm X$', '$\pm Y$' or '$\pm Z$'. For the atom defined by `_atom_local_axes_atom_label`, whose nuclear position is taken as the origin, local axis 'ax1' is the vector from the origin to atom0, axis 'ax2' is perpendicular to 'ax1' and lies in the plane of 'ax1' and a vector passing through the origin parallel to the vector atom1→atom2 (its positive direction making an acute angle with the vector parallel to atom1→atom2), and a right-handed orthonormal vector triplet is formed from the vector product of these two vectors. In most cases, atom1 will be the same as the atom specified by `_atom_local_axes_atom_label`. One or more 'dummy' atoms (with arbitrary labels) may be used in the vector definitions, specified with zero occupancy in the `_atom_site_` description. The values of `*_atom0`, `*_atom1` and `*_atom2` must be identical to values given in the `_atom_site_label` list.
Appears in list containing `_atom_local_axes_atom_label`. **Must** match parent data name `_atom_site_label`. `[atom_local_axes]`

`_atom_local_axes_atom1` (*char*)
Specifies 'atom1' in the definition of a local axis frame. The definition employs three atom-site labels, 'atom0', 'atom1' and 'atom2', and two axis labels, 'ax1' and 'ax2', having values '$\pm X$', '$\pm Y$' or '$\pm Z$'. For the atom defined by `_atom_local_axes_atom_label`, whose nuclear position is taken as the origin, local axis 'ax1' is the vector from the origin to atom0, axis 'ax2' is perpendicular to 'ax1' and lies in the plane of 'ax1' and a vector passing through the origin parallel to the vector atom1→atom2 (its positive direction making an acute angle with the vector parallel to atom1→atom2), and a right-handed orthonormal vector triplet is formed from the vector product of these two vectors. In most cases, atom1 will be the same as the atom specified by `_atom_local_axes_atom_label`. One or more 'dummy' atoms (with arbitrary labels) may be used in the vector definitions, specified with zero occupancy in the `_atom_site_` description. The values of `*_atom0`, `*_atom1` and `*_atom2` must be identical to values given in the `_atom_site_label` list.
Appears in list containing `_atom_local_axes_atom_label`. **Must** match parent data name `_atom_site_label`. `[atom_local_axes]`

`_atom_local_axes_atom2` (*char*)
Specifies 'atom2' in the definition of a local axis frame. The definition employs three atom-site labels, 'atom0', 'atom1' and 'atom2', and two axis labels, 'ax1' and 'ax2', having values '$\pm X$', '$\pm Y$' or '$\pm Z$'. For the atom defined by `_atom_local_axes_atom_label`, whose nuclear position is taken as the origin, local axis 'ax1'

Affiliation: PAUL R. MALLINSON, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, Scotland.

is the vector from the origin to atom0, axis 'ax2' is perpendicular to 'ax1' and lies in the plane of 'ax1' and a vector passing through the origin parallel to the vector atom1→atom2 (its positive direction making an acute angle with the vector parallel to atom1→atom2), and a right-handed orthonormal vector triplet is formed from the vector product of these two vectors. In most cases, atom1 will be the same as the atom specified by **_atom_local_axes_atom_label**. One or more 'dummy' atoms (with arbitrary labels) may be used in the vector definitions, specified with zero occupancy in the **_atom_site_** description. The values of ***_atom0**, ***_atom1** and ***_atom2** must be identical to values given in the **_atom_site_label** list.

Appears in list containing **_atom_local_axes_atom_label**. **Must** match parent data name **_atom_site_label**. **[atom_local_axes]**

_atom_local_axes_ax1 (*char*)

Specifies 'ax1' in the definition of a local axis frame. The definition employs three atom-site labels, 'atom0', 'atom1' and 'atom2', and two axis labels, 'ax1' and 'ax2', having values '$\pm X$', '$\pm Y$' or '$\pm Z$'. For the atom defined by **_atom_local_axes_atom_label**, whose nuclear position is taken as the origin, local axis 'ax1' is the vector from the origin to atom0, axis 'ax2' is perpendicular to 'ax1' and lies in the plane of 'ax1' and a vector passing through the origin parallel to the vector atom1→atom2 (its positive direction making an acute angle with the vector parallel to atom1→atom2), and a right-handed orthonormal vector triplet is formed from the vector product of these two vectors. In most cases, atom1 will be the same as the atom specified by **_atom_local_axes_atom_label**. One or more 'dummy' atoms (with arbitrary labels) may be used in the vector definitions, specified with zero occupancy in the **_atom_site_** description. The values of ***_atom0**, ***_atom1** and ***_atom2** must be identical to values given in the **_atom_site_label** list.

Appears in list containing **_atom_local_axes_atom_label**.

The data value must be one of the following:

x X y Y z Z +x +X +y +Y +z +Z -x -X -y -Y -z -Z. **[atom_local_axes]**

_atom_local_axes_ax2 (*char*)

Specifies 'ax2' in the definition of a local axis frame. The definition employs three atom-site labels, 'atom0', 'atom1' and 'atom2', and two axis labels, 'ax1' and 'ax2', having values '$\pm X$', '$\pm Y$' or '$\pm Z$'. For the atom defined by **_atom_local_axes_atom_label**, whose nuclear position is taken as the origin, local axis 'ax1' is the vector from the origin to atom0, axis 'ax2' is perpendicular to 'ax1' and lies in the plane of 'ax1' and a vector passing through the origin parallel to the vector atom1→atom2 (its positive direction making an acute angle with the vector parallel to atom1→atom2), and a right-handed orthonormal vector triplet is formed from the vector product of these two vectors. In most cases, atom1 will be the same as the atom specified by **_atom_local_axes_atom_label**. One or more 'dummy' atoms (with arbitrary labels) may be used in the vector definitions, specified with zero occupancy in the **_atom_site_** description. The values of ***_atom0**, ***_atom1** and ***_atom2** must be identical to values given in the **_atom_site_label** list.

Appears in list containing **_atom_local_axes_atom_label**.

The data value must be one of the following:

x X y Y z Z +x +X +y +Y +z +Z -x -X -y -Y -z -Z. **[atom_local_axes]**

ATOM_RHO_MULTIPOLE

This category contains information about the multipole coefficients used to describe the electron density. High-resolution X-ray diffraction methods enable the determination of the electron density distribution in crystal lattices and molecules, which in turn allows for a characterization of chemical interactions (Coppens, 1997; Koritsanszky & Coppens, 2001). This is accomplished by the construction of a mathematical model of the charge density in a crystal and then by fitting the parameters of such a model to the experimental pattern of diffracted X-rays. The model on which this dictionary is based is the so-called multipole formalism proposed by Hansen & Coppens (1978). In this model, the electron density in a crystal is described by a sum of aspherical 'pseudoatoms' where the pseudoatom density has the form defined in the **_atom_rho_multipole_*** items. Each pseudoatom density consists of terms representing the core density, the spherical part of the valence density and the deviation of the valence density from sphericity. The continuous electron density in the crystal is then modelled as a sum of atom-centred charge distributions. Once the experimental electron density has been established, the 'atoms in molecules' theory of Bader (1990) provides tools for the interpretation of the density distribution in terms of its topological properties.

References: Bader, R. F. W. (1990). *Atoms in molecules: a quantum theory*. Oxford University Press. Coppens, P. (1997). *X-ray charge densities and chemical bonding*. Oxford University Press. Hansen, N. K. & Coppens, P. (1978). *Acta Cryst.* A**34**, 909–921. Koritsanszky, T. S. & Coppens, P. (2001). *Chem. Rev*. **101**, 1583–1621.

*Example 1 – Multipole coefficients for the nickel ion in [Ni(H$_3$L)][NO$_3$][PF$_6$], [H$_3$L = N,N′,N′′-tris(2-hydroxy-3-methylbutyl)-1,4,7-triazacyclononane] [G. T. Smith et al. (1997). J. Am. Chem. Soc. **119**, 5028–5034].*

```
loop_
_atom_rho_multipole_atom_label
_atom_rho_multipole_coeff_Pv
_atom_rho_multipole_coeff_P00
_atom_rho_multipole_coeff_P11
_atom_rho_multipole_coeff_P1-1
_atom_rho_multipole_coeff_P10
_atom_rho_multipole_coeff_P20
_atom_rho_multipole_coeff_P21
_atom_rho_multipole_coeff_P2-1
_atom_rho_multipole_coeff_P22
_atom_rho_multipole_coeff_P2-2
_atom_rho_multipole_coeff_P30
_atom_rho_multipole_coeff_P31
_atom_rho_multipole_coeff_P3-1
_atom_rho_multipole_coeff_P32
_atom_rho_multipole_coeff_P3-2
_atom_rho_multipole_coeff_P33
_atom_rho_multipole_coeff_P3-3
_atom_rho_multipole_coeff_P40
_atom_rho_multipole_coeff_P41
_atom_rho_multipole_coeff_P4-1
_atom_rho_multipole_coeff_P42
_atom_rho_multipole_coeff_P4-2
_atom_rho_multipole_coeff_P43
_atom_rho_multipole_coeff_P4-3
_atom_rho_multipole_coeff_P44
_atom_rho_multipole_coeff_P4-4
_atom_rho_multipole_kappa
_atom_rho_multipole_kappa_prime0
_atom_rho_multipole_kappa_prime1
_atom_rho_multipole_kappa_prime2
_atom_rho_multipole_kappa_prime3
_atom_rho_multipole_kappa_prime4
Ni2+(1)  2.38(4)  0.32(4)  0.00  0.00 -0.02(1)
         0.00(2)  0.00     0.00  0.00  0.00     -0.08(1)
         0.00     0.00     0.00  0.00  0.06(1)  -0.04(1)
         0.05(1)  0.00     0.00  0.00  0.00     -0.20(1)
         0.08(1)  0.00     0.00
         1.04(1)  0.44(1)  0.44  1.15(4) 0.44  1.15
```

_atom_rho_multipole_atom_label (*char*)
This item is used to identify the atom whose electron density is described with an atom in the ATOM_SITE category. Its value must be identical to one of the values in the **_atom_site_label** list.
Appears in list as essential element of loop structure. **Must** match parent data name **_atom_site_label**. **[atom_rho_multipole]**

_atom_rho_multipole_coeff_Pc
_atom_rho_multipole_coeff_Pv
_atom_rho_multipole_coeff_P00
_atom_rho_multipole_coeff_P10
_atom_rho_multipole_coeff_P11
_atom_rho_multipole_coeff_P1-1
_atom_rho_multipole_coeff_P20
_atom_rho_multipole_coeff_P21
_atom_rho_multipole_coeff_P2-1
_atom_rho_multipole_coeff_P22
_atom_rho_multipole_coeff_P2-2
_atom_rho_multipole_coeff_P30
_atom_rho_multipole_coeff_P31
_atom_rho_multipole_coeff_P3-1
_atom_rho_multipole_coeff_P32
_atom_rho_multipole_coeff_P3-2
_atom_rho_multipole_coeff_P33
_atom_rho_multipole_coeff_P3-3
_atom_rho_multipole_coeff_P40
_atom_rho_multipole_coeff_P41
_atom_rho_multipole_coeff_P4-1
_atom_rho_multipole_coeff_P42
_atom_rho_multipole_coeff_P4-2
_atom_rho_multipole_coeff_P43
_atom_rho_multipole_coeff_P4-3
_atom_rho_multipole_coeff_P44
_atom_rho_multipole_coeff_P4-4 (*numb, su*)
Specifies the multipole population coefficients $P(l,m)$ for the atom identified in **_atom_rho_multipole_atom_label**. The multipoles are defined with respect to the local axes specified in the ATOM_LOCAL_AXES category. The coefficients refer to the multipole formalism described by Hansen & Coppens [1978, equation (2)] which gives the electron density at position vector **r** with respect to an atomic nucleus as

$$\rho(\mathbf{r}) = P_c\rho_{\rm core}(\mathbf{r}) + P_v\kappa^3\rho_{\rm valence}(\kappa r) + \sum\{\kappa'(l)^3 R(\kappa'(l),l,\mathbf{r})\}\sum\{P(l,m)d(l,m,\theta,\varphi)\},$$

where P_c = **_atom_rho_multipole_coeff_Pc**, P_v = **_atom_rho_multipole_coeff_Pv**, $P(0,0)$ = **_atom_rho_multipole_coeff_P00**, $P_c+P_v+P(0,0) = Z$ (the atomic number) for a neutral atom, κ = **_atom_rho_multipole_kappa**, $\kappa'(l)$ = **_atom_rho_multipole_kappa_prime[l]** and $d(l,m,\theta,\varphi)$ is the spherical harmonic of order l,m at the position (θ,φ) with respect to spherical coordinates centred on the atom. The summations are performed over the index ranges $0 \le l \le l_{\rm max}, -l \le m \le l$, respectively, where $l_{\rm max}$ is the highest order of multipole applied. The spherical coordinates are related to the local Cartesian axes defined in category ATOM_LOCAL_AXES, z is the polar axis from which the angle θ is measured, and the angle φ is measured from the x axis in the xy plane with the y axis having a value of $\varphi = +90°$. $R(\kappa'(l),l,\mathbf{r})$ is defined in the **_atom_rho_multipole_radial_*** items. $\rho_{\rm core}(\mathbf{r})$ and $\rho_{\rm valence}(\kappa r)$ are the spherical core and valence densities, respectively. They are obtained from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974). They are also the Fourier transforms of the X-ray scattering factors given in **_atom_rho_multipole_scat_core** and **_atom_rho_multipole_scat_valence.**

References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478. Hansen, N. K. & Coppens, P. (1978). *Acta Cryst.* A**34**, 909–921.
Appears in list containing **_atom_rho_multipole_atom_label**. **[atom_rho_multipole]**

_atom_rho_multipole_configuration (*char*)
This item defines the electronic configuration of the atom given in **_atom_rho_multipole_atom_label** as free text.
Appears in list containing **_atom_rho_multipole_atom_label**. **[atom_rho_multipole]**

_atom_rho_multipole_core_source (*char*),
This item gives the source of the orbital exponents and expansion coefficients used to obtain the spherical core density of the atom defined in **_atom_rho_multipole_atom_label**. Alternatively, the core density may be obtained as described in the **_atom_rho_multipole_scat_core** item.
References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478.
Appears in list containing **_atom_rho_multipole_atom_label**.
Example:
```
; Clementi, E. & Roetti, C. (1974). At. Data Nucl. Data
  Tables, 14, 177-478.
;
```
[atom_rho_multipole]

_atom_rho_multipole_kappa
_atom_rho_multipole_kappa_prime0
_atom_rho_multipole_kappa_prime1
_atom_rho_multipole_kappa_prime2
_atom_rho_multipole_kappa_prime3
_atom_rho_multipole_kappa_prime4 (*numb, su*)
Gives the radial function expansion–contraction coefficients (κ = **_atom_rho_multipole_kappa** and $\kappa'(l)$ = **_atom_rho_multipole_kappa_prime[l]**) for the atom specified in **_atom_rho_multipole_atom_label**. The coefficients refer to the multipole formalism described by Hansen & Coppens [1978, equation (2)], which gives the electron density at position vector **r** with respect to an atomic nucleus as

$$\rho(\mathbf{r}) = P_c\rho_{\rm core}(\mathbf{r}) + P_v\kappa^3\rho_{\rm valence}(\kappa r) + \sum\{\kappa'(l)^3 R(\kappa'(l),l,\mathbf{r})\}\sum\{P(l,m)d(l,m,\theta,\varphi)\},$$

where P_c = **_atom_rho_multipole_coeff_Pc**, P_v = **_atom_rho_multipole_coeff_Pv**, $P(0,0)$ = **_atom_rho_multipole_coeff_P00**, $P_c + P_v + P(0,0) = Z$ (the atomic number) for a neutral atom, $P(l,m)$ = **_atom_rho_multipole_coeff_P[lm]** and $d(l,m,\theta,\varphi)$ is the spherical harmonic of order l,m at the position (θ,φ) with respect to spherical coordinates centred on the atom. The spherical coordinates are related to the local Cartesian axes defined in category ATOM_LOCAL_AXES, z is the polar axis from which the angle θ is measured, and the angle φ is measured from the x axis in the xy plane with the y axis having a value of $\varphi = +90°$. $R(\kappa'(l),l,\mathbf{r})$ is defined in the **_atom_rho_multipole_radial_*** items. $\rho_{\rm core}(\mathbf{r})$ and $\rho_{\rm valence}(\kappa r)$ are the spherical core and valence densities, respectively. They are obtained from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974). They are also the Fourier transforms of the X-ray scattering factors given in **_atom_rho_multipole_scat_core** and **_atom_rho_multipole_scat_valence.** The order, l, of κ' refers to the order of the multipole function, $0 \le l \le 4$. The values of κ' are normally constrained to be equal.

References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478. Hansen, N. K. & Coppens, P. (1978). *Acta Cryst.* A**34**, 909–921.

Appears in list containing `_atom_rho_multipole_atom_label`.

[atom_rho_multipole]

`_atom_rho_multipole_radial_function_type` (*char*)

Specifies the function $R(\kappa'(l),l,\mathbf{r})$ used for the radial dependence of the valence electron density in the multipole formalism described by Hansen & Coppens [1978, equation (2)] which gives the electron density at position vector $\mathbf{r}$ with respect to the nucleus of the atom specified in `_atom_rho_multipole_atom_label` as

$$\rho(\mathbf{r}) = P_c\rho_{\rm core}(\mathbf{r}) + P_v\kappa^3\rho_{\rm valence}(\kappa r) + \sum\{\kappa'(l)^3 R(\kappa'(l),l,\mathbf{r})\}\sum\{P(l,m)d(l,m,\theta,\varphi)\},$$

where P_c = `_atom_rho_multipole_coeff_Pc`, P_v = `_atom_rho_multipole_coeff_Pv`, $P(0,0)$ = `_atom_rho_multipole_coeff_P00`, $P_c + P_v + P(0,0) = Z$ (the atomic number) for a neutral atom, κ = `_atom_rho_multipole_kappa`, $\kappa'(l)$ = `_atom_rho_multipole_kappa_prime[l]`, $P(l,m)$ = `_atom_rho_multipole_coeff_P[lm]` and $d(l,m,\theta,\varphi)$ is the spherical harmonic of order l,m at the position (θ,φ) with respect to spherical coordinates centred on the atom. The summations are performed over the index ranges $0 \le l \le l_{\rm max}, -l \le m \le l$, respectively, where $l_{\rm max}$ is the highest order of multipole applied. The spherical coordinates are related to the local Cartesian axes defined in category ATOM_LOCAL_AXES, z is the polar axis from which the angle θ is measured and the angle φ is measured from the x axis in the xy plane with the y axis having a value of $\varphi = +90°$. $\rho_{\rm core}(\mathbf{r})$ and $\rho_{\rm valence}(\kappa r)$ are the spherical core and valence densities, respectively. They are obtained from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974). They are also the Fourier transforms of the X-ray scattering factors given in `_atom_rho_multipole_scat_core` and `_atom_rho_multipole_scat_valence`.

This item need not be given if a Slater function is used. The parameters of the Slater function should be given using the `_atom_rho_multipole_radial_slater_*` items.

References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478. Hansen, N. K. & Coppens, P. (1978). *Acta Cryst.* A**34**, 909–921.

Appears in list containing `_atom_rho_multipole_atom_label`.

[atom_rho_multipole]

`_atom_rho_multipole_radial_slater_n0`
`_atom_rho_multipole_radial_slater_zeta0`
`_atom_rho_multipole_radial_slater_n1`
`_atom_rho_multipole_radial_slater_zeta1`
`_atom_rho_multipole_radial_slater_n2`
`_atom_rho_multipole_radial_slater_zeta2`
`_atom_rho_multipole_radial_slater_n3`
`_atom_rho_multipole_radial_slater_zeta3`
`_atom_rho_multipole_radial_slater_n4`
`_atom_rho_multipole_radial_slater_zeta4` (*numb, su*)

These items are used when the radial dependence of the electron density, $R(\kappa'(l),l,\mathbf{r})$, of the atom specified in `_atom_rho_multipole_atom_label` is expressed as a Slater-type function [Hansen & Coppens (1978), equation (3)]:

$$R(\kappa'(l),l,\mathbf{r}) = \{\zeta(l)^{n(l)+3}/[n(l)+2]!\}(\kappa'(l)r)^{n(l)}\exp(-\kappa'(l)\zeta(l)r),$$

where $\kappa'(l)$ = `_atom_rho_multipole_kappa_prime[l]`, $n(l)$ = `_atom_rho_multipole_radial_slater_n[l]` and $\zeta(l)$ = `_atom_rho_multipole_slater_zeta[l]`. $R(\kappa'(l),l,\mathbf{r})$ appears in the multipole formalism described by Hansen & Coppens [1978, equation (2)] which gives the electron density at position vector $\mathbf{r}$ with respect to an atomic nucleus as

$$\rho(\mathbf{r}) = P_c\rho_{\rm core}(\mathbf{r}) + P_v\kappa^3\rho_{\rm valence}(\kappa r) + \sum\{\kappa'(l)^3 R(\kappa'(l),l,\mathbf{r})\}\sum\{P(l,m)d(l,m,\theta,\varphi)\},$$

where P_c = `_atom_rho_multipole_coeff_Pc`, P_v = `_atom_rho_multipole_coeff_Pv`, $P(0,0)$ = `_atom_rho_multipole_coeff_P00`, $P_c + P_v + P(0,0) = Z$ (the atomic number) for a neutral atom, κ = `_atom_rho_multipole_kappa`, $\kappa'(l)$ = `_atom_rho_multipole_kappa_prime[l]`, $P(l,m)$ = `_atom_rho_multipole_coeff_P[lm]` and $d(l,m,\theta,\varphi)$ is the spherical harmonic of order l,m at the position (θ,φ) with respect to spherical coordinates centred on the atom. The spherical coordinates are related to the local Cartesian axes defined in category ATOM_LOCAL_AXES, z is the polar axis from which the angle θ is measured, and the angle φ is measured from the x axis in the xy plane with the y axis having a value of $\varphi = +90°$. $\rho_{\rm core}(\mathbf{r})$ and $\rho_{\rm valence}(\kappa r)$ are the spherical core and valence densities, respectively. They are obtained from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974). They are also the Fourier transforms of the X-ray scattering factors given in `_atom_rho_multipole_scat_core` and `_atom_rho_multipole_scat_valence`.

References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478. Hansen, N. K. & Coppens, P. (1978). *Acta Cryst.* A**34**, 909–921.

Appears in list containing `_atom_rho_multipole_atom_label`.

[atom_rho_multipole]

`_atom_rho_multipole_scat_core` (*char*)

This item gives the scattering factor for the core electrons of the atom specified in `_atom_rho_multipole_atom_label` as a function of $\sin(\theta)/\lambda$. The text should contain only a table of two columns, the first giving the value of $\sin(\theta)/\lambda$, the second giving the X-ray scattering factor at this point in reciprocal space. The atomic core scattering factors are used in least-squares fitting of the items in `_atom_rho_multipole_coeff_`* and `_atom_rho_multipole_kappa_`* to experimental X-ray structure factors [see for example Coppens (1997)]. This item enables them to be supplied in the form of a numerical table. Normally, they originate from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974).

References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478. Coppens, P. (1997). *X-ray charge densities and chemical bonding.* Oxford University Press.

Appears in list containing `_atom_rho_multipole_atom_label`.

[atom_rho_multipole]

`_atom_rho_multipole_scat_valence` (*char*)

This item gives the scattering factor for the valence electrons of the atom specified in `_atom_rho_multipole_atom_label` as a function of $\sin(\theta)/\lambda$. The text should contain only a table of two columns, the first giving the value of $\sin(\theta)/\lambda$, the second giving the X-ray scattering factor at this point in reciprocal space. The atomic valence scattering factors are used in least-squares fitting of the items in `_atom_rho_multipole_coeff_`* and `_atom_rho_multipole_kappa_`* to experimental X-ray structure factors [see for example Coppens (1997)]. This item enables them to be supplied in the form of a numerical table. Normally, they originate from atomic orbital analytic wavefunctions such as those tabulated by Clementi & Roetti (1974).

References: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478. Coppens, P. (1997). *X-ray charge densities and chemical bonding.* Oxford University Press.

Appears in list containing **_atom_rho_multipole_atom_label**.

[atom_rho_multipole]

_atom_rho_multipole_valence_source (*char*)

This item gives the source of the orbital exponents and expansion coefficients used to obtain the spherical valence density of the atom defined in **_atom_rho_multipole_atom_label**. Alternatively, the valence density may be obtained as described in the **_atom_rho_multipole_scat_valence** item.

Reference: Clementi, E. & Roetti, C. (1974). *At. Data Nucl. Data Tables*, **14**, 177–478.

Appears in list containing **_atom_rho_multipole_atom_label**.

Example:

```
; Clementi, E. & Roetti, C. (1974). At. Data Nucl. Data
  Tables, 14, 177-478.
;
```

[atom_rho_multipole]

ATOM_SITE

Data items in the ATOM_SITE category record details about the atom sites in a crystal structure, such as the positional coordinates, atomic displacement parameters, magnetic moments and directions.

_atom_site_label (*char*)

The **_atom_site_label** is a unique identifier for a particular site in the crystal, and is fully defined in the core CIF dictionary. The child data names itemized here are in addition to those in the core dictionary.

Appears in list as essential element of loop structure. May match child data name(s):
_atom_local_axes_atom0, **_atom_local_axes_atom1**, **_atom_local_axes_atom2**, **_atom_local_axes_atom_label**, **_atom_rho_multipole_atom_label**.

[atom_site]

4.5. Macromolecular dictionary (mmCIF)

BY P. M. D. FITZGERALD, J. D. WESTBROOK, P. E. BOURNE, B. MCMAHON, K. D. WATENPAUGH AND H. M. BERMAN

This is version 2.0.09 of the macromolecular CIF dictionary (mmCIF). The philosophy behind this dictionary and the history of its development are described in Chapter 1.1. A detailed commentary on the use of the dictionary is given in Chapter 3.6.

Category groups

atom_group	Categories that describe the properties of atoms.
audit_group	Categories that describe dictionary maintenance and identification.
cell_group	Categories that describe the unit cell.
chemical_group	Categories that describe chemical properties and nomenclature.
chem_comp_group	Categories that describe components of chemical structure.
chem_link_group	Categories that describe links between components of chemical structure.
citation_group	Categories that provide bibliographic references.
computing_group	Categories that describe the computational details of the experiment.
compliance_group	Categories that are included in this dictionary specifically to comply with previous dictionaries.
database_group	Categories that hold references to entries in databases that contain related information.
diffrn_group	Categories that describe details of the diffraction experiment.
entity_group	Categories that describe chemical entities.
entry_group	Categories that pertain to the entire data block.
exptl_group	Categories that hold details of the experimental conditions.
geom_group	Categories that hold details of molecular and crystal geometry.
iucr_group	Categories that are used for manuscript submission and internal processing by the staff of the International Union of Crystallography.
pdb_group	Categories that pertain to the file-format or data-processing codes used by the Protein Data Bank.
phasing_group	Categories that describe phasing.
refine_group	Categories that describe refinement.
refln_group	Categories that describe the details of reflection measurements.
struct_group	Categories that contain details about the crystallographic structure.
symmetry_group	Categories that describe symmetry information.

Affiliations: PAULA M. D. FITZGERALD, Merck Research Laboratories, Rahway, New Jersey, USA; JOHN D. WESTBROOK, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ, USA; PHILLIP E. BOURNE, Research Collaboratory for Structural Bioinformatics, San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0537, USA; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England; KEITH D. WATENPAUGH, retired; formerly Structural, Analytical and Medicinal Chemistry, Pharmacia Corporation, Kalamazoo, Michigan, USA; HELEN M. BERMAN, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ, USA.

ATOM_SITE

Data items in the ATOM_SITE category record details about the atom sites in a macromolecular crystal structure, such as the positional coordinates, atomic displacement parameters, magnetic moments and directions. The data items for describing anisotropic atomic displacement factors are only used if the corresponding items are not given in the ATOM_SITE_ANISOTROP category.

Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_site.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_site.group_PDB
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_seq_id
_atom_site.label_alt_id
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.footnote_id
_atom_site.auth_seq_id
_atom_site.id
ATOM N  N   VAL  A  11  .  25.369  30.691  11.795  1.00
     17.93  .  11   1
ATOM C  CA  VAL  A  11  .  25.970  31.965  12.332  1.00
     17.75  .  11   2
ATOM C  C   VAL  A  11  .  25.569  32.010  13.808  1.00
     17.83  .  11   3
ATOM O  O   VAL  A  11  .  24.735  31.190  14.167  1.00
     17.53  .  11   4
ATOM C  CB  VAL  A  11  .  25.379  33.146  11.540  1.00
     17.66  .  11   5
ATOM C  CG1 VAL  A  11  .  25.584  33.034  10.030  1.00
     18.86  .  11   6
ATOM C  CG2 VAL  A  11  .  23.933  33.309  11.872  1.00
     17.12  .  11   7
ATOM N  N   THR  A  12  .  26.095  32.930  14.590  1.00
     18.97  4  12   8
ATOM C  CA  THR  A  12  .  25.734  32.995  16.032  1.00
     19.80  4  12   9
ATOM C  C   THR  A  12  .  24.695  34.106  16.113  1.00
     20.92  4  12  10
ATOM O  O   THR  A  12  .  24.869  35.118  15.421  1.00
     21.84  4  12  11
ATOM C  CB  THR  A  12  .  26.911  33.346  17.018  1.00
     20.51  4  12  12
ATOM O  OG1 THR  A  12  3  27.946  33.921  16.183  0.50
     20.29  4  12  13
```

```
ATOM O  OG1 THR  A  12  4  27.769  32.142  17.103  0.50
     20.59  4  12  14
ATOM C  CG2 THR  A  12  3  27.418  32.181  17.878  0.50
     20.47  4  12  15
ATOM C  CG2 THR  A  12  4  26.489  33.778  18.426  0.50
     20.00  4  12  16
ATOM N  N   ILE  A  13  .  23.664  33.855  16.884  1.00
     22.08  .  13  17
ATOM C  CA  ILE  A  13  .  22.623  34.850  17.093  1.00
     23.44  .  13  18
ATOM C  C   ILE  A  13  .  22.657  35.113  18.610  1.00
     25.77  .  13  19
ATOM O  O   ILE  A  13  .  23.123  34.250  19.406  1.00
     26.28  .  13  20
ATOM C  CB  ILE  A  13  .  21.236  34.463  16.492  1.00
     22.67  .  13  21
ATOM C  CG1 ILE  A  13  .  20.478  33.469  17.371  1.00
     22.14  .  13  22
ATOM C  CG2 ILE  A  13  .  21.357  33.986  15.016  1.00
     21.75  .  13  23
# - - - - data truncated for brevity - - - -
HETATM C C1 APS  C   .  1   4.171  29.012   7.116  0.58
     17.27  1 300  101
HETATM C C2 APS  C   .  1   4.949  27.758   6.793  0.58
     16.95  1 300  102
HETATM O O3 APS  C   .  1   4.800  26.678   7.393  0.58
     16.85  1 300  103
HETATM N N4 APS  C   .  1   5.930  27.841   5.869  0.58
     16.43  1 300  104
# - - - - data truncated for brevity - - - -
```

_atom_site.adp_type (*code*)
_atom_site_adp_type (*cif_core.dic 2.3*)
A standard code used to describe the type of atomic displacement parameters used for the site.
Related item: **_atom_site.thermal_displace_type** (alternate).
The data value must be one of the following:

Uani	anisotropic U^{ij}
Uiso	isotropic U
Uovl	overall U
Umpe	multipole expansion U
Bani	anisotropic B^{ij}
Biso	isotropic B
Bovl	overall B

[atom_site]

_atom_site.aniso_B[1][1] (*float, su*)
The [1][1] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j(B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.
Related items: **_atom_site.aniso_B[1][1]_esd** (associated esd),
_atom_site.aniso_U[1][1] (conversion constant),
_atom_site_anisotrop.U[1][1] (conversion constant),
_atom_site.aniso_U[1][1] (alternate exclusive),
_atom_site_anisotrop.B[1][1] (alternate exclusive),
_atom_site_anisotrop.U[1][1] (alternate exclusive). **[atom_site]**

_atom_site.aniso_B[1][1]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_B[1][1].**
Related items: **_atom_site.aniso_B[1][1]** (associated value),
_atom_site.aniso_U[1][1]_esd (conversion constant),
_atom_site_anisotrop.U[1][1]_esd (conversion constant),
_atom_site.aniso_U[1][1]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][1]_esd (alternate exclusive),
_atom_site_anisotrop.U[1][1]_esd (alternate exclusive). **[atom_site]**

_atom_site.aniso_B[1][2] (*float, su*)
The [1][2] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j(B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.
Related items: **_atom_site.aniso_B[1][2]_esd** (associated esd),
_atom_site.aniso_U[1][2] (conversion constant),
_atom_site_anisotrop.U[1][2] (conversion constant),
_atom_site.aniso_U[1][2] (alternate exclusive),
_atom_site_anisotrop.B[1][2] (alternate exclusive),
_atom_site_anisotrop.U[1][2] (alternate exclusive). **[atom_site]**

_atom_site.aniso_B[1][2]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_B[1][2].**
Related items: **_atom_site.aniso_B[1][2]** (associated value),
_atom_site.aniso_U[1][2]_esd (conversion constant),
_atom_site_anisotrop.U[1][2]_esd (conversion constant),
_atom_site.aniso_U[1][2]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][2]_esd (alternate exclusive),
_atom_site_anisotrop.U[1][2]_esd (alternate exclusive). **[atom_site]**

_atom_site.aniso_B[1][3] (*float, su*)
The [1][3] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j(B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on

Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: `_atom_site.aniso_B[1][3]_esd` (associated esd), `_atom_site.aniso_U[1][3]` (conversion constant), `_atom_site_anisotrop.U[1][3]` (conversion constant), `_atom_site.aniso_U[1][3]` (alternate exclusive), `_atom_site_anisotrop.B[1][3]` (alternate exclusive), `_atom_site_anisotrop.U[1][3]` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[1][3]_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_atom_site.aniso_B[1][3]`.

Related items: `_atom_site.aniso_B[1][3]` (associated value), `_atom_site.aniso_U[1][3]_esd` (conversion constant), `_atom_site_anisotrop.U[1][3]_esd` (conversion constant), `_atom_site.aniso_U[1][3]_esd` (alternate exclusive), `_atom_site_anisotrop.B[1][3]_esd` (alternate exclusive), `_atom_site_anisotrop.U[1][3]_esd` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[2][2]` (*float, su*)

The [2][2] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j(B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: `_atom_site.aniso_B[2][2]_esd` (associated esd), `_atom_site.aniso_U[2][2]` (conversion constant), `_atom_site_anisotrop.U[2][2]` (conversion constant), `_atom_site.aniso_U[2][2]` (alternate exclusive), `_atom_site_anisotrop.B[2][2]` (alternate exclusive), `_atom_site_anisotrop.U[2][2]` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[2][2]_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_atom_site.aniso_B[2][2]`.

Related items: `_atom_site.aniso_B[2][2]` (associated value), `_atom_site.aniso_U[2][2]_esd` (conversion constant), `_atom_site_anisotrop.U[2][2]_esd` (conversion constant), `_atom_site.aniso_U[2][2]_esd` (alternate exclusive), `_atom_site_anisotrop.B[2][2]_esd` (alternate exclusive), `_atom_site_anisotrop.U[2][2]_esd` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[2][3]` (*float, su*)

The [2][3] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j(B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: `_atom_site.aniso_B[2][3]_esd` (associated esd), `_atom_site.aniso_U[2][3]` (conversion constant), `_atom_site_anisotrop.U[2][3]` (conversion constant), `_atom_site.aniso_U[2][3]` (alternate exclusive), `_atom_site_anisotrop.B[2][3]` (alternate exclusive), `_atom_site_anisotrop.U[2][3]` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[2][3]_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_atom_site.aniso_B[2][3]`.

Related items: `_atom_site.aniso_B[2][3]` (associated value), `_atom_site.aniso_U[2][3]_esd` (conversion constant), `_atom_site_anisotrop.U[2][3]_esd` (conversion constant), `_atom_site.aniso_U[2][3]_esd` (alternate exclusive), `_atom_site_anisotrop.B[2][3]_esd` (alternate exclusive), `_atom_site_anisotrop.U[2][3]_esd` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[3][3]` (*float, su*)

The [3][3] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j(B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: `_atom_site.aniso_B[3][3]_esd` (associated esd), `_atom_site.aniso_U[3][3]` (conversion constant), `_atom_site_anisotrop.U[3][3]` (conversion constant), `_atom_site.aniso_U[3][3]` (alternate exclusive), `_atom_site_anisotrop.B[3][3]` (alternate exclusive), `_atom_site_anisotrop.U[3][3]` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_B[3][3]_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_atom_site.aniso_B[3][3]`.

Related items: `_atom_site.aniso_B[3][3]` (associated value), `_atom_site.aniso_U[3][3]_esd` (conversion constant), `_atom_site_anisotrop.U[3][3]_esd` (conversion constant), `_atom_site.aniso_U[3][3]_esd` (alternate exclusive), `_atom_site_anisotrop.B[3][3]_esd` (alternate exclusive), `_atom_site_anisotrop.U[3][3]_esd` (alternate exclusive). **[atom_site]**

`_atom_site.aniso_ratio` (*float*)

Ratio of the maximum to minimum principal axes of displacement (thermal) ellipsoids.

The permitted range is $[1.0, \infty)$.

Related item: `_atom_site_anisotrop.ratio` (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[1][1] (*float, su*)
The [1][1] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2 \sum_i \Big[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site.aniso_U[1][1]_esd** (associated esd),
_atom_site.aniso_B[1][1] (conversion constant),
_atom_site_anisotrop.B[1][1] (conversion constant),
_atom_site.aniso_B[1][1] (alternate exclusive),
_atom_site_anisotrop.B[1][1] (alternate exclusive),
_atom_site_anisotrop.U[1][1] (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[1][1]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_U[1][1].**

Related items: **_atom_site.aniso_U[1][1]** (associated value),
_atom_site.aniso_B[1][1]_esd (conversion constant),
_atom_site_anisotrop.B[1][1]_esd (conversion constant),
_atom_site.aniso_B[1][1]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][1]_esd (alternate exclusive),
_atom_site_anisotrop.U[1][1]_esd (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[1][2] (*float, su*)
The [1][2] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2 \sum_i \Big[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site.aniso_U[1][2]_esd** (associated esd),
_atom_site.aniso_B[1][2] (conversion constant),
_atom_site_anisotrop.B[1][2] (conversion constant),
_atom_site.aniso_B[1][2] (alternate exclusive),
_atom_site_anisotrop.B[1][2] (alternate exclusive),
_atom_site_anisotrop.U[1][2] (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[1][2]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_U[1][2].**

Related items: **_atom_site.aniso_U[1][2]** (associated value),
_atom_site.aniso_B[1][2]_esd (conversion constant),
_atom_site_anisotrop.B[1][2]_esd (conversion constant),
_atom_site.aniso_B[1][2]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][2]_esd (alternate exclusive),
_atom_site_anisotrop.U[1][2]_esd (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[1][3] (*float, su*)
The [1][3] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2 \sum_i \Big[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site.aniso_U[1][3]_esd** (associated esd),
_atom_site.aniso_B[1][3] (conversion constant),
_atom_site_anisotrop.B[1][3] (conversion constant),
_atom_site.aniso_B[1][3] (alternate exclusive),
_atom_site_anisotrop.B[1][3] (alternate exclusive),
_atom_site_anisotrop.U[1][3] (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[1][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_U[1][3].**

Related items: **_atom_site.aniso_U[1][3]** (associated value),
_atom_site.aniso_B[1][3]_esd (conversion constant),
_atom_site_anisotrop.B[1][3]_esd (conversion constant),
_atom_site.aniso_B[1][3]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][3]_esd (alternate exclusive),
_atom_site_anisotrop.U[1][3]_esd (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[2][2] (*float, su*)
The [2][2] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2 \sum_i \Big[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site.aniso_U[2][2]_esd** (associated esd),
_atom_site.aniso_B[2][2] (conversion constant),
_atom_site_anisotrop.B[2][2] (conversion constant),
_atom_site.aniso_B[2][2] (alternate exclusive),
_atom_site_anisotrop.B[2][2] (alternate exclusive),
_atom_site_anisotrop.U[2][2] (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[2][2]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_U[2][2].**

Related items: **_atom_site.aniso_U[2][2]** (associated value),
_atom_site.aniso_B[2][2]_esd (conversion constant),
_atom_site_anisotrop.B[2][2]_esd (conversion constant),
_atom_site.aniso_B[2][2]_esd (alternate exclusive),
_atom_site_anisotrop.B[2][2]_esd (alternate exclusive),
_atom_site_anisotrop.U[2][2]_esd (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[2][3] (*float, su*)
The [2][3] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2 \sum_i \Big[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site.aniso_U[2][3]_esd** (associated esd), **_atom_site.aniso_B[2][3]** (conversion constant), **_atom_site_anisotrop.B[2][3]** (conversion constant), **_atom_site.aniso_B[2][3]** (alternate exclusive), **_atom_site_anisotrop.B[2][3]** (alternate exclusive), **_atom_site_anisotrop.U[2][3]** (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[2][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_U[2][3].**

Related items: **_atom_site.aniso_U[2][3]** (associated value), **_atom_site.aniso_B[2][3]_esd** (conversion constant), **_atom_site_anisotrop.B[2][3]_esd** (conversion constant), **_atom_site.aniso_B[2][3]_esd** (alternate exclusive), **_atom_site_anisotrop.B[2][3]_esd** (alternate exclusive), **_atom_site_anisotrop.U[2][3]_esd** (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[3][3] (*float, su*)
The [3][3] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2 \sum_i \Big[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either B's or U's, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site.aniso_U[3][3]_esd** (associated esd), **_atom_site.aniso_B[3][3]** (conversion constant), **_atom_site_anisotrop.B[3][3]** (conversion constant), **_atom_site.aniso_B[3][3]** (alternate exclusive), **_atom_site_anisotrop.B[3][3]** (alternate exclusive), **_atom_site_anisotrop.U[3][3]** (alternate exclusive). **[atom_site]**

_atom_site.aniso_U[3][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.aniso_U[3][3].**

Related items: **_atom_site.aniso_U[3][3]** (associated value), **_atom_site.aniso_B[3][3]_esd** (conversion constant), **_atom_site_anisotrop.B[3][3]_esd** (conversion constant), **_atom_site.aniso_B[3][3]_esd** (alternate exclusive), **_atom_site_anisotrop.B[3][3]_esd** (alternate exclusive), **_atom_site_anisotrop.U[3][3]_esd** (alternate exclusive). **[atom_site]**

_atom_site.attached_hydrogens (*int*)
_atom_site_attached_hydrogens (*cif_core.dic 2.0.1*)
The number of hydrogen atoms attached to the atom at this site excluding any hydrogen atoms for which coordinates (measured or calculated) are given.

The permitted range is [0, 8].

Examples: '2' (water oxygen), '1' (hydroxyl oxygen), '4' (ammonium nitrogen).

[atom_site]

***_atom_site.auth_asym_id** (*code*)
An alternative identifier for **_atom_site.label_asym_id** that may be provided by an author in order to match the identification used in the publication that describes the structure.

The following item(s) have an equivalent role in their respective categories:

_geom_angle.atom_site_auth_asym_id_1, ***_geom_angle.atom_site_auth_asym_id_2***, ***_geom_angle.atom_site_auth_asym_id_3***, ***_geom_bond.atom_site_auth_asym_id_1***, ***_geom_bond.atom_site_auth_asym_id_2***, ***_geom_contact.atom_site_auth_asym_id_1***, ***_geom_contact.atom_site_auth_asym_id_2***, ***_geom_hbond.atom_site_auth_asym_id_A***, ***_geom_hbond.atom_site_auth_asym_id_D***, ***_geom_hbond.atom_site_auth_asym_id_H***, ***_geom_torsion.atom_site_auth_asym_id_1***, ***_geom_torsion.atom_site_auth_asym_id_2***, ***_geom_torsion.atom_site_auth_asym_id_3***, ***_geom_torsion.atom_site_auth_asym_id_4***, ***_struct_conf.beg_auth_asym_id***, ***_struct_conf.end_auth_asym_id***, ***_struct_conn.ptnr1_auth_asym_id***, ***_struct_conn.ptnr2_auth_asym_id***, ***_struct_mon_nucl.auth_asym_id***, ***_struct_mon_prot.auth_asym_id***, ***_struct_mon_prot_cis.auth_asym_id***, ***_struct_ncs_dom_lim.beg_auth_asym_id***, ***_struct_ncs_dom_lim.end_auth_asym_id***, ***_struct_sheet_range.beg_auth_asym_id***, ***_struct_sheet_range.end_auth_asym_id***, ***_struct_site_gen.auth_asym_id***. **[atom_site]**

_atom_site.auth_atom_id (*atcode*)
An alternative identifier for **_atom_site.label_atom_id** that may be provided by an author in order to match the identification used in the publication that describes the structure.

The following item(s) have an equivalent role in their respective categories:

_geom_angle.atom_site_auth_atom_id_1, ***_geom_angle.atom_site_auth_atom_id_2***, ***_geom_angle.atom_site_auth_atom_id_3***, ***_geom_bond.atom_site_auth_atom_id_1***, ***_geom_bond.atom_site_auth_atom_id_2***, ***_geom_contact.atom_site_auth_atom_id_1***, ***_geom_contact.atom_site_auth_atom_id_2***, ***_geom_hbond.atom_site_auth_atom_id_A***, ***_geom_hbond.atom_site_auth_atom_id_D***, ***_geom_hbond.atom_site_auth_atom_id_H***, ***_geom_torsion.atom_site_auth_atom_id_1***, ***_geom_torsion.atom_site_auth_atom_id_2***, ***_geom_torsion.atom_site_auth_atom_id_3***, ***_geom_torsion.atom_site_auth_atom_id_4***, ***_struct_conn.ptnr1_auth_atom_id***, ***_struct_conn.ptnr2_auth_atom_id***, ***_struct_sheet_hbond.range_1_beg_auth_atom_id***, ***_struct_sheet_hbond.range_1_end_auth_atom_id***, ***_struct_sheet_hbond.range_2_beg_auth_atom_id***, ***_struct_sheet_hbond.range_2_end_auth_atom_id***, ***_struct_site_gen.auth_atom_id***. **[atom_site]**

_atom_site.auth_comp_id (*code*)

An alternative identifier for _atom_site.label_comp_id that may be provided by an author in order to match the identification used in the publication that describes the structure.

The following item(s) have an equivalent role in their respective categories:

_geom_angle.atom_site_auth_comp_id_1,
_geom_angle.atom_site_auth_comp_id_2,
_geom_angle.atom_site_auth_comp_id_3,
_geom_bond.atom_site_auth_comp_id_1,
_geom_bond.atom_site_auth_comp_id_2,
_geom_contact.atom_site_auth_comp_id_1,
_geom_contact.atom_site_auth_comp_id_2,
_geom_hbond.atom_site_auth_comp_id_A,
_geom_hbond.atom_site_auth_comp_id_D,
_geom_hbond.atom_site_auth_comp_id_H,
_geom_torsion.atom_site_auth_comp_id_1,
_geom_torsion.atom_site_auth_comp_id_2,
_geom_torsion.atom_site_auth_comp_id_3,
_geom_torsion.atom_site_auth_comp_id_4,
_struct_conf.beg_auth_comp_id,
_struct_conf.end_auth_comp_id,
_struct_conn.ptnr1_auth_comp_id,
_struct_conn.ptnr2_auth_comp_id,
_struct_mon_nucl.auth_comp_id,
_struct_mon_prot.auth_comp_id,
_struct_mon_prot_cis.auth_comp_id,
_struct_ncs_dom_lim.beg_auth_comp_id,
_struct_ncs_dom_lim.end_auth_comp_id,
_struct_sheet_range.beg_auth_comp_id,
_struct_sheet_range.end_auth_comp_id,
_struct_site_gen.auth_comp_id. **[atom_site]**

_atom_site.auth_seq_id (*code*)

An alternative identifier for _atom_site.label_seq_id that may be provided by an author in order to match the identification used in the publication that describes the structure. Note that this is not necessarily a number, that the values do not have to be positive, and that the value does not have to correspond to the value of _atom_site.label_seq_id. The value of _atom_site.label_seq_id is required to be a sequential list of positive integers. The author may assign values to _atom_site.auth_seq_id in any desired way. For instance, the values may be used to relate this structure to a numbering scheme in a homologous structure, including sequence gaps or insertion codes. Alternatively, a scheme may be used for a truncated polymer that maintains the numbering scheme of the full length polymer. In all cases, the scheme used here must match the scheme used in the publication that describes the structure.

The following item(s) have an equivalent role in their respective categories:

_geom_angle.atom_site_auth_seq_id_1,
_geom_angle.atom_site_auth_seq_id_2,
_geom_angle.atom_site_auth_seq_id_3,
_geom_bond.atom_site_auth_seq_id_1,
_geom_bond.atom_site_auth_seq_id_2,
_geom_contact.atom_site_auth_seq_id_1,
_geom_contact.atom_site_auth_seq_id_2,
_geom_hbond.atom_site_auth_seq_id_A,
_geom_hbond.atom_site_auth_seq_id_D,
_geom_hbond.atom_site_auth_seq_id_H,
_geom_torsion.atom_site_auth_seq_id_1,
_geom_torsion.atom_site_auth_seq_id_2,
_geom_torsion.atom_site_auth_seq_id_3,
_geom_torsion.atom_site_auth_seq_id_4,
_struct_conf.beg_auth_seq_id,
_struct_conf.end_auth_seq_id,
_struct_conn.ptnr1_auth_seq_id,
_struct_conn.ptnr2_auth_seq_id,
_struct_mon_nucl.auth_seq_id,
_struct_mon_prot.auth_seq_id,
_struct_mon_prot_cis.auth_seq_id,
_struct_ncs_dom_lim.beg_auth_seq_id,
_struct_ncs_dom_lim.end_auth_seq_id,
_struct_sheet_hbond.range_1_beg_auth_seq_id,
_struct_sheet_hbond.range_1_end_auth_seq_id,
_struct_sheet_hbond.range_2_beg_auth_seq_id,
_struct_sheet_hbond.range_2_end_auth_seq_id,
_struct_sheet_range.beg_auth_seq_id,
_struct_sheet_range.end_auth_seq_id,
_struct_site_gen.auth_seq_id. **[atom_site]**

_atom_site.B_equiv_geom_mean (*float, su*)

_atom_site_B_equiv_geom_mean (*cif_core.dic 2.0.1*)

Equivalent isotropic atomic displacement parameter, $B_{\rm eq}$, in ångströms squared, calculated as the geometric mean of the anisotropic atomic displacement parameters.

$$B_{\rm eq} = (B_i B_j B_k)^{1/3},$$

where B_n = the principal components of the orthogonalized B^{ij}.

The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

The permitted range is $[0.0, \infty)$.

Related items: _atom_site.B_equiv_geom_mean_esd (associated esd), _atom_site.U_equiv_geom_mean (conversion constant). **[atom_site]**

_atom_site.B_equiv_geom_mean_esd (*float*)

The standard uncertainty (estimated standard deviation) of _atom_site.B_equiv_geom_mean.

Related items: _atom_site.B_equiv_geom_mean (associated value), _atom_site.U_equiv_geom_mean (conversion constant). **[atom_site]**

_atom_site.B_iso_or_equiv (*float, su*)

_atom_site_B_iso_or_equiv (*cif_core.dic 2.0.1*)

Isotropic atomic displacement parameter, or equivalent isotropic atomic displacement parameter, $B_{\rm eq}$, calculated from the anisotropic displacement parameters.

$$B_{\rm eq} = (1/3) \sum_i \Big[\sum_j (B^{ij} A_i A_j a_i^* a_j^*) \Big],$$

where A = the real-space cell lengths and a^* = the reciprocal-space cell lengths; $B^{ij} = 8\pi^2 U^{ij}$.

Reference: Fischer, R. X. & Tillmanns, E. (1988). *Acta Cryst.* C**44**, 775–776.

The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: _atom_site.B_iso_or_equiv_esd (associated esd), _atom_site.U_iso_or_equiv (conversion constant). **[atom_site]**

_atom_site.B_iso_or_equiv_esd (*float*)

The standard uncertainty (estimated standard deviation) of _atom_site.B_iso_or_equiv.

Related items: _atom_site.B_iso_or_equiv (associated value), _atom_site.U_iso_or_equiv_esd (conversion constant). **[atom_site]**

_atom_site.calc_attached_atom (*code*)

_atom_site_calc_attached_atom (*cif_core.dic 2.0.1*)

The _atom_site.id of the atom site to which the 'geometry-calculated' atom site is attached.

[atom_site]

_atom_site.calc_flag (*ucode*)
_atom_site_calc_flag (*cif_core.dic 2.0.1*)
A standard code to signal whether the site coordinates have been determined from the intensities or calculated from the geometry of surrounding sites, or have been assigned dummy values. The abbreviation 'c' may be used in place of 'calc'.
The data value must be one of the following:

d	determined from experimental measurements
calc	calculated from molecular geometry
c	abbreviation for 'calc'
dum	dummy site with meaningless coordinates

[atom_site]

_atom_site.Cartn_x (*float, su*)
_atom_site_Cartn_x (*cif_core.dic 2.0.1*)
The *x* atom-site coordinate in ångströms specified according to a set of orthogonal Cartesian axes related to the cell axes as specified by the description given in **_atom_sites.Cartn_transform_axes**.
Related item: **_atom_site.Cartn_x_esd** (associated esd). **[atom_site]**

_atom_site.Cartn_x_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.Cartn_x**.
Related item: **_atom_site.Cartn_x** (associated value). **[atom_site]**

_atom_site.Cartn_y (*float, su*)
_atom_site_Cartn_y (*cif_core.dic 2.0.1*)
The *y* atom-site coordinate in ångströms specified according to a set of orthogonal Cartesian axes related to the cell axes as specified by the description given in **_atom_sites.Cartn_transform_axes**.
Related item: **_atom_site.Cartn_y_esd** (associated esd). **[atom_site]**

_atom_site.Cartn_y_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.Cartn_y**.
Related item: **_atom_site.Cartn_y** (associated value). **[atom_site]**

_atom_site.Cartn_z (*float, su*)
_atom_site_Cartn_z (*cif_core.dic 2.0.1*)
The *z* atom-site coordinate in ångströms specified according to a set of orthogonal Cartesian axes related to the cell axes as specified by the description given in **_atom_sites.Cartn_transform_axes**.
Related item: **_atom_site.Cartn_z_esd** (associated esd). **[atom_site]**

_atom_site.Cartn_z_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.Cartn_z**.
Related item: **_atom_site.Cartn_z** (associated value). **[atom_site]**

_atom_site.chemical_conn_number
_atom_site_chemical_conn_number (*cif_core.dic 2.0.1*)
This data item is a pointer to **_chemical_conn_atom.number** in the CHEMICAL_CONN_ATOM category.

_atom_site.constraints (*line*)
_atom_site_constraints (*cif_core.dic 2.0.1*)
A description of the constraints applied to parameters at this site during refinement. See also **_atom_site.refinement_flags** and **_refine.ls_number_constraints**.
Example: 'pop=1.0-pop(Zn3)'. **[atom_site]**

_atom_site.details (*text*)
_atom_site_description (*cif_core.dic 2.0.1*)
A description of special aspects of this site. See also **_atom_site.refinement_flags**.
Example: 'Ag/Si disordered'. **[atom_site]**

_atom_site.disorder_assembly (*code*)
_atom_site_disorder_assembly (*cif_core.dic 2.0.1*)
A code which identifies a cluster of atoms that show long-range positional disorder but are locally ordered. Within each such cluster of atoms, **_atom_site.disorder_group** is used to identify the sites that are simultaneously occupied. This field is only needed if there is more than one cluster of disordered atoms showing independent local order.
Note: This data item would not in general be used in a macromolecular data block.

[atom_site]

_atom_site.disorder_group (*code*)
_atom_site_disorder_group (*cif_core.dic 2.0.1*)
A code which identifies a group of positionally disordered atom sites that are locally simultaneously occupied. Atoms that are positionally disordered over two or more sites (*e.g.* the hydrogen atoms of a methyl group that exists in two orientations) can be assigned to two or more groups. Sites belonging to the same group are simultaneously occupied, but those belonging to different groups are not. A minus prefix (*e.g.* '-1') is used to indicate sites disordered about a special position.
Note: This data item would not in general be used in a macromolecular data block.
Where no value is given, the assumed value is '.'. **[atom_site]**

_atom_site.footnote_id
The value of **_atom_site.footnote_id** must match an ID specified by **_atom_sites_footnote.id** in the ATOM_SITES_FOOTNOTE list.

_atom_site.fract_x (*float, su*)
_atom_site_fract_x (*cif_core.dic 2.0.1*)
The *x* coordinate of the atom-site position specified as a fraction of **_cell.length_a**.
Related item: **_atom_site.fract_x_esd** (associated esd). **[atom_site]**

_atom_site.fract_x_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.fract_x**.
Related item: **_atom_site.fract_x** (associated value). **[atom_site]**

_atom_site.fract_y (*float, su*)
_atom_site_fract_y (*cif_core.dic 2.0.1*)
The *y* coordinate of the atom-site position specified as a fraction of **_cell.length_b**.
Related item: **_atom_site.fract_y_esd** (associated esd). **[atom_site]**

_atom_site.fract_y_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.fract_y**.
Related item: **_atom_site.fract_y** (associated value). **[atom_site]**

_atom_site.fract_z (*float, su*)
_atom_site_fract_z (*cif_core.dic 2.0.1*)
The *z* coordinate of the atom-site position specified as a fraction of **_cell.length_c**.
Related item: **_atom_site.fract_z_esd** (associated esd). **[atom_site]**

_atom_site.fract_z_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.fract_z**.
Related item: **_atom_site.fract_z** (associated value). **[atom_site]**

_atom_site.group_PDB (*code*)
The group of atoms to which the atom site belongs. This data item is provided for compatibility with the original Protein Data Bank format, and only for that purpose.
The data value must be one of the following:
ATOM
HETATM **[atom_site]**

*__atom_site.id** (*code*)
_atom_site_label (*cif_core.dic 2.0.1*)
The value of **_atom_site.id** must uniquely identify a record in the ATOM_SITE list. Note that this item need not be a number; it can be any unique identifier. This data item was introduced to provide compatibility between small-molecule and macromolecular CIFs. In a small-molecule CIF, **_atom_site_label** is the identifier for the atom. In a macromolecular CIF, the atom identifier is the aggregate of **_atom_site.label_alt_id**, **_atom_site.label_asym_id**, **_atom_site.label_atom_id**, **_atom_site.label_comp_id** and **_atom_site.label_seq_id**. For the two types of files to be compatible, a formal identifier for the category had to be introduced that was independent of the different modes of identifying the atoms. For compatibility with older CIFs, **_atom_site_label** is aliased to **_atom_site.id**.
The following item(s) have an equivalent role in their respective categories:
_atom_site_anisotrop.id,
_geom_angle.atom_site_id_1,
_geom_angle.atom_site_id_2,
_geom_angle.atom_site_id_3,
_geom_bond.atom_site_id_1,
_geom_bond.atom_site_id_2,
_geom_contact.atom_site_id_1,
_geom_contact.atom_site_id_2,
_geom_hbond.atom_site_id_A,
_geom_hbond.atom_site_id_D,
_geom_hbond.atom_site_id_H,
_geom_torsion.atom_site_id_1,
_geom_torsion.atom_site_id_2,
_geom_torsion.atom_site_id_3,
_geom_torsion.atom_site_id_4.
Examples: '5', 'C12', 'Ca3g28', 'Fe3+17', 'H*251', 'boron2a', 'C_a_phe_83_a_0', 'Zn_Zn_301_A_0'. **[atom_site]**

*__atom_site.label_alt_id**
A component of the identifier for this atom site. For further details, see the definition of the ATOM_SITE_ALT category. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

*__atom_site.label_asym_id**
A component of the identifier for this atom site. For further details, see the definition of the STRUCT_ASYM category. This data item is a pointer to **_struct_asym.id** in the STRUCT_ASYM category.

*__atom_site.label_atom_id**
A component of the identifier for this atom site. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

*__atom_site.label_comp_id**
A component of the identifier for this atom site. This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

*__atom_site.label_entity_id**
This data item is a pointer to **_entity.id** in the ENTITY category.

*__atom_site.label_seq_id**
This data item is a pointer to **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category.

_atom_site.occupancy (*float, su*)
_atom_site_occupancy (*cif_core.dic 2.0.1*)
The fraction of the atom type present at this site. The sum of the occupancies of all the atom types at this site may not significantly exceed 1.0 unless it is a dummy site.
Related item: **_atom_site.occupancy_esd** (associated esd). Where no value is given, the assumed value is '1.0'. **[atom_site]**

_atom_site.occupancy_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.occupancy**.
Related item: **_atom_site.occupancy** (associated value). **[atom_site]**

_atom_site.refinement_flags (*code*)
_atom_site_refinement_flags (*cif_core.dic 2.3*)
A concatenated series of single-letter codes which indicate the refinement restraints or constraints applied to this site. This item should not be used. It has been replaced by **_atom_site.refinement_flags_posn**, *__adp** and *__occupancy**. It is retained in this dictionary only to provide compatibility with old CIFs.
Related items: **_atom_site.refinement_flags_posn** (replaces),
_atom_site.refinement_flags_adp (replaces),
_atom_site.refinement_flags_occupancy (replaces).
The data value must be one of the following:

.	no refinement constraints
S	special-position constraint on site
G	rigid-group refinement of site
R	riding-atom site attached to non-riding atom
D	distance or angle restraint on site
T	thermal displacement constraints
U	$U_{\rm iso}$ or U^{ij} restraint (rigid bond)
P	partial occupancy constraint

[atom_site]

_atom_site.refinement_flags_adp (*code*)
_atom_site_refinement_flags_adp (*cif_core.dic 2.3*)
A code which indicates the refinement restraints or constraints applied to the atomic displacement parameters of this site.
Related item: **_atom_site.refinement_flags_posn** (alternate).
The data value must be one of the following:

.	no constraints on atomic displacement parameters
T	special-position constraints on atomic displacement parameters
U	$U_{\rm iso}$ or U^{ij} restraint (rigid bond)
TU	both constraints applied

[atom_site]

_atom_site.refinement_flags_occupancy (*code*)
_atom_site_refinement_flags_occupancy (*cif_core.dic 2.3*)
A code which indicates that refinement restraints or constraints were applied to the occupancy of this site.
Related item: **_atom_site.refinement_flags_posn** (alternate).
The data value must be one of the following:

.	no constraints on site-occupancy parameters
P	site-occupancy constraint

[atom_site]

_atom_site.refinement_flags_posn (*code*)
_atom_site_refinement_flags_posn (*cif_core.dic 2.3*)
A code which indicates the refinement restraints or constraints applied to the positional coordinates of this site.
Related item: **_atom_site.refinement_flags_posn** (alternate).
The data value must be one of the following:

.	no constraints on positional coordinates
D	distance or angle restraint on positional coordinates
G	rigid-group refinement of positional coordinates
R	riding-atom site attached to non-riding atom
S	special-position constraint on positional coordinates
DG	combination of the above constraints
DR	combination of the above constraints
DS	combination of the above constraints
GR	combination of the above constraints
GS	combination of the above constraints
RS	combination of the above constraints
DGR	combination of the above constraints
DGS	combination of the above constraints
DRS	combination of the above constraints
GRS	combination of the above constraints
DGRS	combination of the above constraints

[atom_site]

_atom_site.restraints (*text*)
_atom_site_restraints (*cif_core.dic 2.0.1*)
A description of restraints applied to specific parameters at this site during refinement. See also **_atom_site.refinement_flags** and **_refine.ls_number_restraints**.
Example: 'restrained to planar ring'. **[atom_site]**

_atom_site.symmetry_multiplicity (*int*)
_atom_site_symmetry_multiplicity (*cif_core.dic 2.0.1*)
The multiplicity of a site due to the space-group symmetry as is given in *International Tables for Crystallography* Vol. A (2002).
The permitted range is [1, 192]. **[atom_site]**

_atom_site.thermal_displace_type (*ucode*)
_atom_site_thermal_displace_type (*cif_core.dic 2.0.1*)
A standard code used to describe the type of atomic displacement parameters used for the site.
The data value must be one of the following:

Uani	anisotropic U^{ij}
Uiso	isotropic U
Uovl	overall U
Umpe	multipole expansion U
Bani	anisotropic B^{ij}
Biso	isotropic B
Bovl	overall B

[atom_site]

***_atom_site.type_symbol**
_atom_site_type_symbol (*cif_core.dic 2.0.1*)
This data item is a pointer to **_atom_type.symbol** in the ATOM_TYPE category.

_atom_site.U_equiv_geom_mean (*float, su*)
_atom_site_U_equiv_geom_mean (*cif_core.dic 2.0.1*)
Equivalent isotropic atomic displacement parameter, $U_{\rm eq}$, in ångströms squared, calculated as the geometric mean of the anisotropic atomic displacement parameters.

$$U_{\rm eq} = (U_i U_j U_k)^{1/3},$$

where U_n = the principal components of the orthogonalized U^{ij}.
The permitted range is [0.0, 10.0].
Related items: **_atom_site.U_equiv_geom_mean_esd** (associated esd), **_atom_site.B_equiv_geom_mean** (conversion constant). **[atom_site]**

_atom_site.U_equiv_geom_mean_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.U_equiv_geom_mean**.
Related items: **_atom_site.U_equiv_geom_mean** (associated value), **_atom_site.B_equiv_geom_mean** (conversion constant). **[atom_site]**

_atom_site.U_iso_or_equiv (*float, su*)
_atom_site_U_iso_or_equiv (*cif_core.dic 2.0.1*)
Isotropic atomic displacement parameter, or equivalent isotropic atomic displacement parameter, $U_{\rm eq}$, calculated from anisotropic atomic displacement parameters.

$$U_{\rm eq} = (1/3) \sum_i \left[\sum_j (U^{ij} A_i A_j a_i^* a_j^*) \right],$$

where A = the real-space cell lengths and a^* = the reciprocal-space cell lengths.

Reference: Fischer, R. X. & Tillmanns, E. (1988). *Acta Cryst.* C**44**, 775–776.

The permitted range is [0.0, 10.0].
Related items: **_atom_site.U_iso_or_equiv_esd** (associated esd), **_atom_site.B_iso_or_equiv** (conversion constant). **[atom_site]**

_atom_site.U_iso_or_equiv_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site.U_iso_or_equiv**.
Related items: **_atom_site.U_iso_or_equiv** (associated value), **_atom_site.B_iso_or_equiv_esd** (conversion constant). **[atom_site]**

_atom_site.Wyckoff_symbol (*line*)
_atom_site_Wyckoff_symbol (*cif_core.dic 2.0.1*)
The Wyckoff symbol (letter) as listed in the space-group tables of *International Tables for Crystallography* Vol. A (2002).

[atom_site]

ATOM_SITE_ANISOTROP

Data items in the ATOM_SITE_ANISOTROP category record details about anisotropic displacement parameters. If the ATOM_SITE_ANISOTROP category is used for storing these data, the corresponding ATOM_SITE data items are not used.
Category group(s): **inclusive_group**, **atom_group**
Category key(s): **_atom_site_anisotrop.id**

*Example 1 – based on NDB structure BDL005 of Holbrook, Dickerson & Kim [Acta Cryst. (1985), B**41**, 255–262].*

```
loop_
_atom_site_anisotrop.id
_atom_site_anisotrop.type_symbol
_atom_site_anisotrop.U[1][1]
_atom_site_anisotrop.U[1][2]
_atom_site_anisotrop.U[1][3]
_atom_site_anisotrop.U[2][2]
_atom_site_anisotrop.U[2][3]
_atom_site_anisotrop.U[3][3]
 1  O   8642   4866   7299   -342   -258  -1427
 2  C   5174   4871   6243  -1885  -2051  -1377
 3  C   6202   5020   4395  -1130   -556   -632
 4  O   4224   4700   5046   1105   -161    345
 5  C   8684   4688   4171  -1850   -433   -292
 6  O  11226   5255   3532   -341   2685   1328
 7  C  10214   2428   5614  -2610  -1940    902
 8  C   4590   3488   5827    751   -770    986
 9  N   5014   4434   3447    -17  -1593    539
# ----   abbreviated  ----
```

_atom_site_anisotrop.B[1][1] (*float, su*)
_atom_site_aniso_B_11 (*cif_core.dic 2.0.1*)
The [1][1] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: **_atom_site_anisotrop.B[1][1]_esd** (associated esd), **_atom_site.aniso_U[1][1]** (conversion constant), **_atom_site_anisotrop.U[1][1]** (conversion constant), **_atom_site.aniso_B[1][1]** (alternate exclusive), **_atom_site.aniso_U[1][1]** (alternate exclusive), **_atom_site_anisotrop.U[1][1]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[1][1]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.B[1][1].**

Related items: **_atom_site_anisotrop.B[1][1]** (associated value), **_atom_site.aniso_U[1][1]_esd** (conversion constant), **_atom_site_anisotrop.U[1][1]_esd** (conversion constant), **_atom_site.aniso_B[1][1]_esd** (alternate exclusive), **_atom_site.aniso_U[1][1]_esd** (alternate exclusive), **_atom_site_anisotrop.U[1][1]_esd** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[1][2] (*float, su*)
_atom_site_aniso_B_12 (*cif_core.dic 2.0.1*)
The [1][2] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: **_atom_site_anisotrop.B[1][2]_esd** (associated esd), **_atom_site.aniso_U[1][2]** (conversion constant), **_atom_site_anisotrop.U[1][2]** (conversion constant), **_atom_site.aniso_B[1][2]** (alternate exclusive), **_atom_site.aniso_U[1][2]** (alternate exclusive), **_atom_site_anisotrop.U[1][2]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[1][2]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.B[1][2].**

Related items: **_atom_site_anisotrop.B[1][2]** (associated value), **_atom_site.aniso_U[1][2]_esd** (conversion constant), **_atom_site_anisotrop.U[1][2]_esd** (conversion constant), **_atom_site.aniso_B[1][2]_esd** (alternate exclusive), **_atom_site.aniso_U[1][2]_esd** (alternate exclusive), **_atom_site_anisotrop.U[1][2]_esd** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[1][3] (*float, su*)
_atom_site_aniso_B_13 (*cif_core.dic 2.0.1*)
The [1][3] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: **_atom_site_anisotrop.B[1][3]_esd** (associated esd), **_atom_site.aniso_U[1][3]** (conversion constant), **_atom_site_anisotrop.U[1][3]** (conversion constant), **_atom_site.aniso_B[1][3]** (alternate exclusive), **_atom_site.aniso_U[1][3]** (alternate exclusive), **_atom_site_anisotrop.U[1][3]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[1][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.B[1][3].**

Related items: **_atom_site_anisotrop.B[1][3]** (associated value), **_atom_site.aniso_U[1][3]_esd** (conversion constant), **_atom_site_anisotrop.U[1][3]_esd** (conversion constant), **_atom_site.aniso_B[1][3]_esd** (alternate exclusive), **_atom_site.aniso_U[1][3]_esd** (alternate exclusive), **_atom_site_anisotrop.U[1][3]_esd** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[2][2] (*float, su*)
_atom_site_aniso_B_22 (*cif_core.dic 2.0.1*)
The [2][2] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_i h_j a_i^* a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting

atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: **_atom_site_anisotrop.B[2][2]_esd** (associated esd), **_atom_site.aniso_U[2][2]** (conversion constant), **_atom_site_anisotrop.U[2][2]** (conversion constant), **_atom_site.aniso_B[2][2]** (alternate exclusive), **_atom_site.aniso_U[2][2]** (alternate exclusive), **_atom_site_anisotrop.U[2][2]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[2][2]_esd (*float*)

The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.B[2][2].**

Related items: **_atom_site_anisotrop.B[2][2]** (associated value), **_atom_site.aniso_U[2][2]_esd** (conversion constant), **_atom_site_anisotrop.U[2][2]_esd** (conversion constant), **_atom_site.aniso_B[2][2]_esd** (alternate exclusive), **_atom_site.aniso_U[2][2]_esd** (alternate exclusive), **_atom_site_anisotrop.U[2][2]_esd** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[2][3] (*float, su*)

_atom_site_aniso_B_23 (*cif_core.dic 2.0.1*)

The [2][3] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: **_atom_site_anisotrop.B[2][3]_esd** (associated esd), **_atom_site.aniso_U[2][3]** (conversion constant), **_atom_site_anisotrop.U[2][3]** (conversion constant), **_atom_site.aniso_B[2][3]** (alternate exclusive), **_atom_site.aniso_U[2][3]** (alternate exclusive), **_atom_site_anisotrop.U[2][3]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[2][3]_esd (*float*)

The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.B[2][3].**

Related items: **_atom_site_anisotrop.B[2][3]** (associated value), **_atom_site.aniso_U[2][3]_esd** (conversion constant), **_atom_site_anisotrop.U[2][3]_esd** (conversion constant), **_atom_site.aniso_B[2][3]_esd** (alternate exclusive), **_atom_site.aniso_U[2][3]_esd** (alternate exclusive), **_atom_site_anisotrop.U[2][3]_esd** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[3][3] (*float, su*)

_atom_site_aniso_B_33 (*cif_core.dic 2.0.1*)

The [3][3] element of the anisotropic atomic displacement matrix **B**, which appears in the structure-factor term as

$$T = \exp\Big\{-(1/4)\sum_i\Big[\sum_j (B^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row. The IUCr Commission on Nomenclature recommends against the use of **B** for reporting atomic displacement parameters. **U**, being directly proportional to **B**, is preferred.

Related items: **_atom_site_anisotrop.B[3][3]_esd** (associated esd), **_atom_site.aniso_U[3][3]** (conversion constant), **_atom_site_anisotrop.U[3][3]** (conversion constant), **_atom_site.aniso_B[3][3]** (alternate exclusive), **_atom_site.aniso_U[3][3]** (alternate exclusive), **_atom_site_anisotrop.U[3][3]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.B[3][3]_esd (*float*)

The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.B[3][3].**

Related items: **_atom_site_anisotrop.B[3][3]** (associated value), **_atom_site.aniso_U[3][3]_esd** (conversion constant), **_atom_site_anisotrop.U[3][3]_esd** (conversion constant), **_atom_site.aniso_B[3][3]_esd** (alternate exclusive), **_atom_site.aniso_U[3][3]_esd** (alternate exclusive), **_atom_site_anisotrop.U[3][3]_esd** (alternate exclusive).

[atom_site_anisotrop]

***_atom_site_anisotrop.id**

_atom_site_aniso_label (*cif_core.dic 2.0.1*)

This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

_atom_site_anisotrop.ratio (*float*)

_atom_site_aniso_ratio (*cif_core.dic 2.0.1*)

Ratio of the maximum to minimum principal axes of displacement (thermal) ellipsoids.

The permitted range is $[1.0, \infty)$.

Related item: **_atom_site.aniso_ratio** (alternate exclusive).

[atom_site_anisotrop]

***_atom_site_anisotrop.type_symbol**

_atom_site_aniso_type_symbol (*cif_core.dic 2.0.1*)

This data item is a pointer to **_atom_type.symbol** in the ATOM_TYPE category.

_atom_site_anisotrop.U[1][1] (*float, su*)

_atom_site_aniso_U_11 (*cif_core.dic 2.0.1*)

The [1][1] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\Big\{-2\pi^2\sum_i\Big[\sum_j (U^{ij}h_ih_ja_i^*a_j^*)\Big]\Big\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row.

Related items: **_atom_site_anisotrop.U[1][1]_esd** (associated esd), **_atom_site.aniso_B[1][1]** (conversion constant), **_atom_site_anisotrop.B[1][1]** (conversion constant), **_atom_site.aniso_B[1][1]** (alternate exclusive), **_atom_site.aniso_U[1][1]** (alternate exclusive), **_atom_site_anisotrop.B[1][1]** (alternate exclusive).

[atom_site_anisotrop]

_atom_site_anisotrop.U[1][1]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.U[1][1].**
Related items: **_atom_site_anisotrop.U[1][1]** (associated value),
_atom_site.aniso_B[1][1]_esd (conversion constant),
_atom_site_anisotrop.B[1][1]_esd (conversion constant),
_atom_site.aniso_B[1][1]_esd (alternate exclusive),
_atom_site.aniso_U[1][1]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][1]_esd (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[1][2] (*float, su*)
_atom_site_aniso_U_12 (*cif_core.dic 2.0.1*)
The [1][2] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\left\{-2\pi^2 \sum_i \left[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\right]\right\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row.
Related items: **_atom_site_anisotrop.U[1][2]_esd** (associated esd),
_atom_site.aniso_B[1][2] (conversion constant),
_atom_site_anisotrop.B[1][2] (conversion constant),
_atom_site.aniso_B[1][2] (alternate exclusive),
_atom_site.aniso_U[1][2] (alternate exclusive),
_atom_site_anisotrop.B[1][2] (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[1][2]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.U[1][2].**
Related items: **_atom_site_anisotrop.U[1][2]** (associated value),
_atom_site.aniso_B[1][2]_esd (conversion constant),
_atom_site_anisotrop.B[1][2]_esd (conversion constant),
_atom_site.aniso_B[1][2]_esd (alternate exclusive),
_atom_site.aniso_U[1][2]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][2]_esd (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[1][3] (*float, su*)
_atom_site_aniso_U_13 (*cif_core.dic 2.0.1*)
The [1][3] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\left\{-2\pi^2 \sum_i \left[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\right]\right\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row.
Related items: **_atom_site_anisotrop.U[1][3]_esd** (associated esd),
_atom_site.aniso_B[1][3] (conversion constant),
_atom_site_anisotrop.B[1][3] (conversion constant),
_atom_site.aniso_B[1][3] (alternate exclusive),
_atom_site.aniso_U[1][3] (alternate exclusive),
_atom_site_anisotrop.B[1][3] (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[1][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.U[1][3].**
Related items: **_atom_site_anisotrop.U[1][3]** (associated value),
_atom_site.aniso_B[1][3]_esd (conversion constant),
_atom_site_anisotrop.B[1][3]_esd (conversion constant),
_atom_site.aniso_B[1][3]_esd (alternate exclusive),
_atom_site.aniso_U[1][3]_esd (alternate exclusive),
_atom_site_anisotrop.B[1][3]_esd (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[2][2] (*float, su*)
_atom_site_aniso_U_22 (*cif_core.dic 2.0.1*)
The [2][2] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\left\{-2\pi^2 \sum_i \left[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\right]\right\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row.
Related items: **_atom_site_anisotrop.U[2][2]_esd** (associated esd),
_atom_site.aniso_B[2][2] (conversion constant),
_atom_site_anisotrop.B[2][2] (conversion constant),
_atom_site.aniso_B[2][2] (alternate exclusive),
_atom_site.aniso_U[2][2] (alternate exclusive),
_atom_site_anisotrop.B[2][2] (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[2][2]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.U[2][2].**
Related items: **_atom_site_anisotrop.U[2][2]** (associated value),
_atom_site.aniso_B[2][2]_esd (conversion constant),
_atom_site_anisotrop.B[2][2]_esd (conversion constant),
_atom_site.aniso_B[2][2]_esd (alternate exclusive),
_atom_site.aniso_U[2][2]_esd (alternate exclusive),
_atom_site_anisotrop.B[2][2]_esd (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[2][3] (*float, su*)
_atom_site_aniso_U_23 (*cif_core.dic 2.0.1*)
The [2][3] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\left\{-2\pi^2 \sum_i \left[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\right]\right\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row.
Related items: **_atom_site_anisotrop.U[2][3]_esd** (associated esd),
_atom_site.aniso_B[2][3] (conversion constant),
_atom_site_anisotrop.B[2][3] (conversion constant),
_atom_site.aniso_B[2][3] (alternate exclusive),
_atom_site.aniso_U[2][3] (alternate exclusive),
_atom_site_anisotrop.B[2][3] (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[2][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.U[2][3].**
Related items: **_atom_site_anisotrop.U[2][3]** (associated value),
_atom_site.aniso_B[2][3]_esd (conversion constant),
_atom_site_anisotrop.B[2][3]_esd (conversion constant),
_atom_site.aniso_B[2][3]_esd (alternate exclusive),
_atom_site.aniso_U[2][3]_esd (alternate exclusive),
_atom_site_anisotrop.B[2][3]_esd (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[3][3] (*float, su*)
_atom_site_aniso_U_33 (*cif_core.dic 2.0.1*)
The [3][3] element of the standard anisotropic atomic displacement matrix **U**, which appears in the structure-factor term as

$$T = \exp\left\{-2\pi^2 \sum_i \left[\sum_j (U^{ij} h_i h_j a_i^* a_j^*)\right]\right\},$$

where h = the Miller indices and a^* = the reciprocal-space cell lengths.

These matrix elements may appear with atomic coordinates in the ATOM_SITE category, or they may appear in the separate ATOM_SITE_ANISOTROP category, but they may not appear in both places. Similarly, anisotropic displacements may appear as either *B*'s or *U*'s, but not as both. The unique elements of the real symmetric matrix are entered by row.
Related items: **_atom_site_anisotrop.U[3][3]_esd** (associated esd),
_atom_site.aniso_B[3][3] (conversion constant),
_atom_site_anisotrop.B[3][3] (conversion constant),
_atom_site.aniso_B[3][3] (alternate exclusive),
_atom_site.aniso_U[3][3] (alternate exclusive),
_atom_site_anisotrop.B[3][3] (alternate exclusive).
[atom_site_anisotrop]

_atom_site_anisotrop.U[3][3]_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_atom_site_anisotrop.U[3][3].**
Related items: **_atom_site_anisotrop.U[3][3]** (associated value),
_atom_site.aniso_B[3][3]_esd (conversion constant),
_atom_site_anisotrop.B[3][3]_esd (conversion constant),
_atom_site.aniso_B[3][3]_esd (alternate exclusive),
_atom_site.aniso_U[3][3]_esd (alternate exclusive),
_atom_site_anisotrop.B[3][3]_esd (alternate exclusive).
[atom_site_anisotrop]

ATOM_SITES

Data items in the ATOM_SITES category record details about the crystallographic cell and cell transformations, which are common to all atom sites.
Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_sites.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_atom_sites.entry_id                    '5HVP'
_atom_sites.Cartn_transform_axes
     'c along z, astar along x, b along y'
_atom_sites.Cartn_transf_matrix[1][1] 58.39
_atom_sites.Cartn_transf_matrix[1][2]  0.00
_atom_sites.Cartn_transf_matrix[1][3]  0.00
_atom_sites.Cartn_transf_matrix[2][1]  0.00
_atom_sites.Cartn_transf_matrix[2][2] 86.70
_atom_sites.Cartn_transf_matrix[2][3]  0.00
_atom_sites.Cartn_transf_matrix[3][1]  0.00
_atom_sites.Cartn_transf_matrix[3][2]  0.00
_atom_sites.Cartn_transf_matrix[3][3] 46.27
_atom_sites.Cartn_transf_vector[1]     0.00
_atom_sites.Cartn_transf_vector[2]     0.00
_atom_sites.Cartn_transf_vector[3]     0.00
```

_atom_sites.Cartn_transf_matrix[1][1] (*float*)
_atom_sites_Cartn_tran_matrix_11 (*cif_core.dic 2.0.1*)
The [1][1] element of the 3×3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3×1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[1][2] (*float*)
_atom_sites_Cartn_tran_matrix_12 (*cif_core.dic 2.0.1*)
The [1][2] element of the 3×3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3×1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[1][3] (*float*)
_atom_sites_Cartn_tran_matrix_13 (*cif_core.dic 2.0.1*)
The [1][3] element of the 3×3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3×1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[2][1] (*float*)
_atom_sites_Cartn_tran_matrix_21 (*cif_core.dic 2.0.1*)
The [2][1] element of the 3×3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3×1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[2][2] (*float*)
_atom_sites_Cartn_tran_matrix_22 (*cif_core.dic 2.0.1*)
The [2][2] element of the 3 × 3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[2][3] (*float*)
_atom_sites_Cartn_tran_matrix_23 (*cif_core.dic 2.0.1*)
The [2][3] element of the 3 × 3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[3][1] (*float*)
_atom_sites_Cartn_tran_matrix_31 (*cif_core.dic 2.0.1*)
The [3][1] element of the 3 × 3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[3][2] (*float*)
_atom_sites_Cartn_tran_matrix_32 (*cif_core.dic 2.0.1*)
The [3][2] element of the 3 × 3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_matrix[3][3] (*float*)
_atom_sites_Cartn_tran_matrix_33 (*cif_core.dic 2.0.1*)
The [3][3] element of the 3 × 3 matrix used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.Cartn_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_vector[1] (*float*)
_atom_sites_Cartn_tran_vector_1 (*cif_core.dic 2.0.1*)
The [1] element of the three-element vector used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The rotation matrix is defined in **_atom_sites.Cartn_transf_matrix**[][].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_vector[2] (*float*)
_atom_sites_Cartn_tran_vector_2 (*cif_core.dic 2.0.1*)
The [2] element of the three-element vector used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The rotation matrix is defined in **_atom_sites.Cartn_transf_matrix**[][].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transf_vector[3] (*float*)
_atom_sites_Cartn_tran_vector_3 (*cif_core.dic 2.0.1*)
The [3] element of the three-element vector used to transform fractional coordinates in the ATOM_SITE category to Cartesian coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The rotation matrix is defined in **_atom_sites.Cartn_transf_matrix**[][].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{fractional}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.Cartn_transform_axes (*text*)
_atom_sites_Cartn_transform_axes (*cif_core.dic 2.0.1*)
A description of the relative alignment of the crystal cell axes to the Cartesian orthogonal axes as applied in the transformation matrix **_atom_sites.Cartn_transf_matrix**[][].
Example: 'a parallel to x; b in the plane of y and z'.
[atom_sites]

* **_atom_sites.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_atom_sites.fract_transf_matrix[1][1] (*float*)
_atom_sites_fract_tran_matrix_11 (*cif_core.dic 2.0.1*)
The [1][1] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[1][2] (*float*)
_atom_sites_fract_tran_matrix_12 (*cif_core.dic 2.0.1*)
The [1][2] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[1][3] (*float*)
_atom_sites_fract_tran_matrix_13 (*cif_core.dic 2.0.1*)
The [1][3] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[2][1] (*float*)
_atom_sites_fract_tran_matrix_21 (*cif_core.dic 2.0.1*)
The [2][1] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[2][2] (*float*)
_atom_sites_fract_tran_matrix_22 (*cif_core.dic 2.0.1*)
The [2][2] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[2][3] (*float*)
_atom_sites_fract_tran_matrix_23 (*cif_core.dic 2.0.1*)
The [2][3] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[3][1] (*float*)
_atom_sites_fract_tran_matrix_31 (*cif_core.dic 2.0.1*)
The [3][1] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[3][2] (*float*)
_atom_sites_fract_tran_matrix_32 (*cif_core.dic 2.0.1*)
The [3][2] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_matrix[3][3] (*float*)
_atom_sites_fract_tran_matrix_33 (*cif_core.dic 2.0.1*)
The [3][3] element of the 3 × 3 matrix used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 1 translation is defined in **_atom_sites.fract_transf_vector**[].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_vector[1] (*float*)
_atom_sites_fract_tran_vector_1 (*cif_core.dic 2.0.1*)
The [1] element of the three-element vector used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 3 rotation is defined in **_atom_sites.fract_transf_matrix**[][].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_vector[2] (*float*)
_atom_sites_fract_tran_vector_2 (*cif_core.dic 2.0.1*)
The [2] element of the three-element vector used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 3 rotation is defined in **_atom_sites.fract_transf_matrix**[][].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.fract_transf_vector[3] (*float*)
_atom_sites_fract_tran_vector_3 (*cif_core.dic 2.0.1*)
The [3] element of the three-element vector used to transform Cartesian coordinates in the ATOM_SITE category to fractional coordinates in the same category. The axial alignments of this transformation are described in **_atom_sites.Cartn_transform_axes**. The 3 × 3 rotation is defined in **_atom_sites.fract_transf_matrix**[][].

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{fractional}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

[atom_sites]

_atom_sites.solution_hydrogens (*ucode*)
_atom_sites_solution_hydrogens (*cif_core.dic 2.0.1*)
This code identifies the method used to locate the hydrogen atoms.

Note: This data item would not in general be used in a macromolecular data block.

The data value must be one of the following:

`difmap`	difference Fourier map
`vecmap`	real-space vector search
`heavy`	heavy-atom method
`direct`	structure-invariant direct methods
`geom`	inferred from neighbouring sites
`disper`	anomalous-dispersion techniques
`isomor`	isomorphous structure methods

[atom_sites]

_atom_sites.solution_primary (*ucode*)
_atom_sites_solution_primary (*cif_core.dic 2.0.1*)
This code identifies the method used to locate the initial atom sites.

Note: This data item would not in general be used in a macromolecular data block.

The data value must be one of the following:

`difmap`	difference Fourier map
`vecmap`	real-space vector search
`heavy`	heavy-atom method
`direct`	structure-invariant direct methods
`geom`	inferred from neighbouring sites
`disper`	anomalous-dispersion techniques
`isomor`	isomorphous structure methods

[atom_sites]

_atom_sites.solution_secondary (*ucode*)
_atom_sites_solution_secondary (*cif_core.dic 2.0.1*)
This code identifies the method used to locate the non-hydrogen-atom sites not found by **_atom_sites.solution_primary**.

Note: This data item would not in general be used in a macromolecular data block.

The data value must be one of the following:

`difmap`	difference Fourier map
`vecmap`	real-space vector search
`heavy`	heavy-atom method
`direct`	structure-invariant direct methods
`geom`	inferred from neighbouring sites
`disper`	anomalous-dispersion techniques
`isomor`	isomorphous structure methods

[atom_sites]

_atom_sites.special_details (*text*)
_atom_sites_special_details (*cif_core.dic 2.3*)
Additional information about the atomic coordinates not coded elsewhere in the CIF.

[atom_sites]

ATOM_SITES_ALT

Data items in the ATOM_SITES_ALT category record details about the structural ensembles that should be generated from atom sites or groups of atom sites that are modelled in alternative conformations in this data block.

Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_sites_alt.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_sites_alt.id
_atom_sites_alt.details
 .
; Atom sites with the alternative ID set to null are not
  modeled in alternative conformations
;
 1
; Atom sites with the alternative ID set to 1 have been
  modeled in alternative conformations with respect to atom
  sites marked with alternative ID 2. The conformations of
  amino-acid side chains and solvent atoms with alternative
  ID set to 1 correlate with the conformation of the
  inhibitor marked with alternative ID 1. They have been
  given an occupancy of 0.58 to match the occupancy assigned
  to the inhibitor.
;
```

```
 2
; Atom sites with the alternative ID set to 2 have been
  modeled in alternative conformations with respect to atom
  sites marked with alternative ID 1. The conformations of
  amino-acid side chains and solvent atoms with alternative
  ID set to 2 correlate with the conformation of the
  inhibitor marked with alternative ID 2. They have been
  given an occupancy of 0.42 to match the occupancy assigned
  to the inhibitor.
;
 3
; Atom sites with the alternative ID set to 3 have been
  modeled in alternative conformations with respect to
  atoms marked with alternative ID 4. The conformations of
  amino-acid side chains and solvent atoms with alternative
  ID set to 3 do not correlate with the conformation of the
  inhibitor. These atom sites have arbitrarily been given
  an occupancy of 0.50.
;
 4
; Atom sites with the alternative ID set to 4 have been
  modeled in alternative conformations with respect to
  atoms marked with alternative ID 3. The conformations of
  amino-acid side chains and solvent atoms with alternative
  ID set to 4 do not correlate with the conformation of the
  inhibitor. These atom sites have arbitrarily been given
  an occupancy of 0.50.
;
```

_atom_sites_alt.details (*text*)

A description of special aspects of the modelling of atoms in alternative conformations.

[atom_sites_alt]

***_atom_sites_alt.id** (*code*)

The value of **_atom_sites_alt.id** must uniquely identify a record in the ATOM_SITES_ALT list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_atom_site.label_alt_id,
_atom_sites_alt_gen.alt_id,
_geom_angle.atom_site_label_alt_id_1,
_geom_angle.atom_site_label_alt_id_2,
_geom_angle.atom_site_label_alt_id_3,
_geom_bond.atom_site_label_alt_id_1,
_geom_bond.atom_site_label_alt_id_2,
_geom_contact.atom_site_label_alt_id_1,
_geom_contact.atom_site_label_alt_id_2,
_geom_hbond.atom_site_label_alt_id_A,
_geom_hbond.atom_site_label_alt_id_D,
_geom_hbond.atom_site_label_alt_id_H,
_geom_torsion.atom_site_label_alt_id_1,
_geom_torsion.atom_site_label_alt_id_2,
_geom_torsion.atom_site_label_alt_id_3,
_geom_torsion.atom_site_label_alt_id_4,
_struct_conn.ptnr1_label_alt_id,
_struct_conn.ptnr2_label_alt_id,
_struct_mon_nucl.label_alt_id,
_struct_mon_prot.label_alt_id,
_struct_mon_prot_cis.label_alt_id,
_struct_ncs_dom_lim.beg_label_alt_id,
_struct_ncs_dom_lim.end_label_alt_id,
_struct_site_gen.label_alt_id.

Examples: '`orientation 1`', '`molecule abc`'. **[atom_sites_alt]**

ATOM_SITES_ALT_ENS

Data items in the ATOM_SITES_ALT_ENS category record details about the ensemble structure generated from atoms with various alternative conformation IDs.

Category group(s): **inclusive_group**
atom_group

Category key(s): **_atom_sites_alt_ens.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_sites_alt_ens.id
_atom_sites_alt_ens.details
 'Ensemble 1-A'
; The inhibitor binds to the enzyme in two, roughly twofold
  symmetric alternative conformations.

  This conformational ensemble includes the more populated
  conformation of the inhibitor (ID=1) and the amino-acid
  side chains and solvent structure that correlate with this
  inhibitor conformation.

  Also included are one set (ID=3) of side chains with
  alternative conformations when the conformations are not
  correlated with the inhibitor conformation.
;
 'Ensemble 1-B'
; The inhibitor binds to the enzyme in two, roughly twofold
  symmetric alternative conformations.

  This conformational ensemble includes the more populated
  conformation of the inhibitor (ID=1) and the amino-acid
  side chains and solvent structure that correlate with
  this inhibitor conformation.

  Also included are one set (ID=4) of side chains with
  alternative conformations when the conformations are not
  correlated with the inhibitor conformation.
;
 'Ensemble 2-A'
; The inhibitor binds to the enzyme in two, roughly twofold
  symmetric alternative conformations.

  This conformational ensemble includes the less populated
  conformation of the inhibitor (ID=2) and the amino-acid
  side chains and solvent structure that correlate with this
  inhibitor conformation.

  Also included are one set (ID=3) of side chains with
  alternative conformations when the conformations are not
  correlated with the inhibitor conformation.
;
 'Ensemble 2-B'
; The inhibitor binds to the enzyme in two, roughly twofold
  symmetric alternative conformations.

  This conformational ensemble includes the less populated
  conformation of the inhibitor (ID=2) and the amino-acid
  side chains and solvent structure that correlate with this
  inhibitor conformation.

  Also included are one set (ID=4) of side chains with
  alternative conformations when the conformations are not
  correlated with the inhibitor conformation.
;
```

_atom_sites_alt_ens.details (*text*)

A description of special aspects of the ensemble structure generated from atoms with various alternative IDs.

[atom_sites_alt_ens]

***_atom_sites_alt_ens.id** (*code*)

The value of **_atom_sites_alt_ens.id** must uniquely identify a record in the ATOM_SITES_ALT_ENS list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_atom_sites_alt_gen.ens_id. **[atom_sites_alt_ens]**

ATOM_SITES_ALT_GEN

Data items in the ATOM_SITES_ALT_GEN category record details about the interpretation of multiple conformations in the structure.

Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_sites_alt_gen.ens_id**
_atom_sites_alt_gen.alt_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_sites_alt_gen.ens_id
_atom_sites_alt_gen.alt_id
 'Ensemble 1-A'  .
 'Ensemble 1-A'  1
 'Ensemble 1-A'  3
 'Ensemble 1-B'  .
 'Ensemble 1-B'  1
 'Ensemble 1-B'  4
 'Ensemble 2-A'  .
 'Ensemble 2-A'  2
 'Ensemble 2-A'  3
 'Ensemble 2-B'  .
 'Ensemble 2-B'  2
 'Ensemble 2-B'  4
```

*_atom_sites_alt_gen.alt_id

This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

*_atom_sites_alt_gen.ens_id

This data item is a pointer to **_atom_sites_alt_ens.id** in the ATOM_SITES_ALT_ENS category.

ATOM_SITES_FOOTNOTE

Data items in the ATOM_SITES_FOOTNOTE category record detailed comments about an atom site or a group of atom sites.

Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_sites_footnote.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_sites_footnote.id
_atom_sites_footnote.text
  1
; The inhibitor binds to the enzyme in two alternative
  orientations. The two orientations have been assigned
  alternative IDs *1* and *2*.
;
  2
; Side chains of these residues adopt alternative
  orientations that correlate with the alternative
  orientations of the inhibitor.
  Side chains with alternative ID *1* and occupancy 0.58
  correlate with inhibitor orientation *1*.
  Side chains with alternative ID *2* and occupancy 0.42
  correlate with inhibitor orientation *2*.
;
  3
; The positions of these water molecules correlate with
  the alternative orientations of the inhibitor.
  Water molecules with alternative ID *1* and occupancy 0.58
  correlate with inhibitor orientation *1*.
  Water molecules with alternative ID *2* and occupancy 0.42
  correlate with inhibitor orientation *2*.
;
  4
; Side chains of these residues adopt alternative
  orientations that do not correlate with the alternative
  orientation of the inhibitor.
;
  5
; The positions of these water molecules correlate with
  alternative orientations of amino-acid side chains that
  do not correlate with alternative orientations of the
  inhibitor.
;
```

*_atom_sites_footnote.id (*code*)

A code that identifies the footnote.

The following item(s) have an equivalent role in their respective categories:

_atom_site.footnote_id.

Examples: 'a', 'b', '1', '2'. **[atom_sites_footnote]**

_atom_sites_footnote.text (*text*)

The text of the footnote. Footnotes are used to describe an atom site or a group of atom sites in the ATOM_SITE list. For example, footnotes may be used to indicate atoms for which the electron density is very weak, or atoms for which static disorder has been modelled.

[atom_sites_footnote]

ATOM_TYPE

Data items in the ATOM_TYPE category record details about the properties of the atoms that occupy the atom sites, such as the atomic scattering factors.

Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_type.symbol**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_type.symbol
_atom_type.oxidation_number
_atom_type.scat_Cromer_Mann_a1
_atom_type.scat_Cromer_Mann_a2
_atom_type.scat_Cromer_Mann_a3
_atom_type.scat_Cromer_Mann_a4
_atom_type.scat_Cromer_Mann_b1
_atom_type.scat_Cromer_Mann_b2
_atom_type.scat_Cromer_Mann_b3
_atom_type.scat_Cromer_Mann_b4
_atom_type.scat_Cromer_Mann_c
  C   0   2.31000  20.8439    1.02000   10.2075
          1.58860   0.568700  0.865000  51.6512      0.21560
  N   0  12.2126    0.005700  3.13220    9.89330
          2.01250  28.9975    1.16630    0.582600  -11.529
  O   0   3.04850  13.2771    2.28680    5.70110
          1.54630   0.323900  0.867000  32.9089      0.250800
  S   0   6.90530   1.46790   5.20340   22.2151
          1.43790   0.253600  1.58630   56.1720      0.866900
  CL -1  18.2915    0.006600  7.20840    1.17170
          6.53370  19.5424    2.33860   60.4486    -16.378
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_atom_type.symbol
_atom_type.oxidation_number
_atom_type.number_in_cell
_atom_type.scat_dispersion_real
_atom_type.scat_dispersion_imag
_atom_type.scat_source
 C  0  72  .017  .009  International_Tables_Vol_IV_Table_2.2B
 H  0  100  0     0    International_Tables_Vol_IV_Table_2.2B
 O  0  12  .047  .032  International_Tables_Vol_IV_Table_2.2B
 N  0  4   .029  .018  International_Tables_Vol_IV_Table_2.2B
```

_atom_type.analytical_mass_percent (*float*)
_atom_type_analytical_mass_% (*cif_core.dic 2.0.1*)
Mass percentage of this atom type derived from chemical analysis.
The permitted range is $[0.0, \infty)$. **[atom_type]**

_atom_type.description (*text*)
_atom_type_description (*cif_core.dic 2.0.1*)
A description of the atom(s) designated by this atom type. In most cases, this is the element name and oxidation state of a single atom species. For disordered or nonstoichiometric structures it will describe a combination of atom species.
Examples: '`deuterium`', '`0.34Fe+0.66Ni`'. **[atom_type]**

_atom_type.number_in_cell (*int*)
_atom_type_number_in_cell (*cif_core.dic 2.0.1*)
Total number of atoms of this atom type in the unit cell.
The permitted range is $[0, \infty)$. **[atom_type]**

_atom_type.oxidation_number (*int*)
_atom_type_oxidation_number (*cif_core.dic 2.0.1*)
Formal oxidation state of this atom type in the structure.
The permitted range is $[-8, 8]$. Where no value is given, the assumed value is '`0`'.
[atom_type]

_atom_type.radius_bond (*float*)
_atom_type_radius_bond (*cif_core.dic 2.0.1*)
The effective intramolecular bonding radius in ångströms of this atom type.
The permitted range is [0.0, 5.0]. **[atom_type]**

_atom_type.radius_contact (*float*)
_atom_type_radius_contact (*cif_core.dic 2.0.1*)
The effective intermolecular bonding radius in ångströms of this atom type.
The permitted range is [0.0, 5.0]. **[atom_type]**

_atom_type.scat_Cromer_Mann_a1 (*float*)
_atom_type_scat_Cromer_Mann_a1 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient a_1 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_a2 (*float*)
_atom_type_scat_Cromer_Mann_a2 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient a_2 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_a3 (*float*)
_atom_type_scat_Cromer_Mann_a3 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient a_3 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_a4 (*float*)
_atom_type_scat_Cromer_Mann_a4 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient a_4 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_b1 (*float*)
_atom_type_scat_Cromer_Mann_b1 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient b_1 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_b2 (*float*)
_atom_type_scat_Cromer_Mann_b2 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient b_2 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_b3 (*float*)
_atom_type_scat_Cromer_Mann_b3 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient b_3 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_b4 (*float*)
_atom_type_scat_Cromer_Mann_b4 (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient b_4 used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_Cromer_Mann_c (*float*)
_atom_type_scat_Cromer_Mann_c (*cif_core.dic 2.0.1*)
The Cromer–Mann scattering-factor coefficient c used to calculate the scattering factors for this atom type.

References: *International Tables for X-ray Crystallography* (1974). Vol. IV, Table 2.2B; *International Tables for Crystallography* (2004). Vol. C, Tables 6.1.1.4 and 6.1.1.5.

[atom_type]

_atom_type.scat_dispersion_imag (*float*)
_atom_type_scat_dispersion_imag (*cif_core.dic 2.0.1*)
The imaginary component of the anomalous-dispersion scattering factor, f'', in electrons for this atom type and the radiation identified by **_diffrn_radiation_wavelength.id**.

[atom_type]

_atom_type.scat_dispersion_real (*float*)
_atom_type_scat_dispersion_real (*cif_core.dic 2.0.1*)
The real component of the anomalous-dispersion scattering factor, f', in electrons for this atom type and the radiation identified by **_diffrn_radiation_wavelength.id**.

[atom_type]

_atom_type.scat_dispersion_source (*text*)
_atom_type_scat_dispersion_source (*cif_core.dic 2.3*)
Reference to the source of the real and imaginary dispersion corrections for scattering factors used for this atom type.
Example: 'International Tables Vol. IV Table 2.3.1'. **[atom_type]**

_atom_type.scat_length_neutron (*text*)
_atom_type_scat_length_neutron (*cif_core.dic 2.0.1*)
The bound coherent scattering length in femtometres for the atom type at the isotopic composition used for the diffraction experiment.
[atom_type]

_atom_type.scat_source (*text*)
_atom_type_scat_source (*cif_core.dic 2.0.1*)
Reference to the source of the scattering factors or scattering lengths used for this atom type.
Example: 'International Tables Vol. IV Table 2.4.6B'. **[atom_type]**

_atom_type.scat_versus_stol_list (*text*)
_atom_type_scat_versus_stol_list (*cif_core.dic 2.0.1*)
A table of scattering factors as a function of $(\sin\theta)/\lambda$. This table should be well commented to indicate the items present. Regularly formatted lists are strongly recommended.
[atom_type]

***_atom_type.symbol** (*code*)
_atom_type_symbol (*cif_core.dic 2.0.1*)
The code used to identify the atom species (singular or plural) representing this atom type. Normally this code is the element symbol. The code may be composed of any character except an underscore with the additional proviso that digits designate an oxidation state and must be followed by a + or - character.
The following item(s) have an equivalent role in their respective categories:
_atom_site.type_symbol,
_atom_site_anisotrop.type_symbol,
_chemical_conn_atom.type_symbol,
_chem_comp_atom.type_symbol,
_phasing_MIR_der_site.atom_type_symbol.
Examples: 'C', 'Cu2+', 'H(SDS)', 'dummy', 'FeNi'. **[atom_type]**

AUDIT

Data items in the AUDIT category record details about the creation and subsequent updating of the data block. Note that these items apply only to the creation and updating of the data block, and should not be confused with the data items in the JOURNAL category that record different stages in the publication of the material in the data block.
Category group(s): **inclusive_group**
audit_group
Category key(s): **_audit.revision_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_audit.revision_id                 1
_audit.creation_date              '1992-12-08'

_audit.creation_method
; Created by hand from PDB entry 5HVP, from the J. Biol.
  Chem. paper describing this structure and from
  laboratory records
;

_audit.update_record
; 1992-12-09  adjusted to reflect comments from B. McKeever
  1992-12-10  adjusted to reflect comments from H. Berman
  1992-12-12  adjusted to reflect comments from K. Watenpaugh
;
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
_audit.creation_date              1991-03-20
_audit.creation_method    from_xtal_archive_file_using_CIFIO
_audit.update_record
; 1991-04-09  text and data added by Tony Willis.
  1991-04-15  rec'd by co-editor as manuscript HL0007.
  1991-04-17  adjustments based on first referee report.
  1991-04-18  adjustments based on second referee report.
;
```

_audit.creation_date (*yyyy-mm-dd*)
_audit_creation_date (*cif_core.dic 2.0.1*)
A date that the data block was created. The date format is *yyyy-mm-dd*.
Example: '1990-07-12'. **[audit]**

_audit.creation_method (*text*)
_audit_creation_method (*cif_core.dic 2.0.1*)
A description of how data were entered into the data block.
Example: 'spawned by the program QBEE'. **[audit]**

***_audit.revision_id** (*code*)
The value of **_audit.revision_id** must uniquely identify a record in the AUDIT list.
Example: 'rev1'. **[audit]**

_audit.update_record (*text*)
_audit_update_record (*cif_core.dic 2.0.1*)
A record of any changes to the data block. The update format is a date (*yyyy-mm-dd*) followed by a description of the changes. The latest update entry is added to the bottom of this record.
Example: '1990-07-15 Updated by the Co-editor'. **[audit]**

AUDIT_AUTHOR

Data items in the AUDIT_AUTHOR category record details about the author(s) of the data block.
Category group(s): **inclusive_group**
audit_group
Category key(s): **_audit_author.name**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_audit_author.name
_audit_author.address
 'Fitzgerald, Paula M.D.'
; Department of Biophysical Chemistry
  Merck Research Laboratories
  P. O. Box 2000, Ry80M203
  Rahway, New Jersey  07065
  USA
;
 'McKeever, Brian M.'
; Department of Biophysical Chemistry
  Merck Research Laboratories
  P. O. Box 2000, Ry80M203
  Rahway, New Jersey  07065
  USA
;
 'Van Middlesworth, J.F.'
; Department of Biophysical Chemistry
  Merck Research Laboratories
  P. O. Box 2000, Ry80M203
  Rahway, New Jersey  07065
  USA
;
 'Springer, James P.'
; Department of Biophysical Chemistry
  Merck Research Laboratories
  P. O. Box 2000, Ry80M203
  Rahway, New Jersey  07065
  USA
;
```

_audit_author.address (*text*)
_audit_author_address (*cif_core.dic 2.0.1*)

The address of an author of this data block. If there are multiple authors, **_audit_author.address** is looped with **_audit_author.name**.

Example:
```
; Department
  Institute
  Street
  City and postcode
  COUNTRY
;
```
[audit_author]

***_audit_author.name** (*line*)
_audit_author_name (*cif_core.dic 2.0.1*)

The name of an author of this data block. If there are multiple authors, **_audit_author.name** is looped with **_audit_author.address**. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.'. **[audit_author]**

AUDIT_CONFORM

Data items in the AUDIT_CONFORM category describe the dictionary versions against which the data names appearing in the current data block are conformant.

Category group(s): **inclusive_group**
audit_group
Category key(s): **_audit_conform.dict_name**
_audit_conform.dict_version

Example 1 – any file conforming to the current CIF core dictionary.

```
_audit_conform.dict_name        cif_core.dic
_audit_conform.dict_version     2.3.1
_audit_conform.dict_location
     ftp://ftp.iucr.org/pub/cif_core.2.3.1.dic
```

_audit_conform.dict_location (*text*)
_audit_conform_dict_location (*cif_core.dic 2.0.1*)

A file name or uniform resource locator (URL) for the dictionary to which the current data block conforms.

[audit_conform]

_audit_conform.dict_name (*text*)
_audit_conform_dict_name (*cif_core.dic 2.0.1*)

The string identifying the highest-level dictionary defining data names used in this file.

[audit_conform]

_audit_conform.dict_version (*text*)
_audit_conform_dict_version (*cif_core.dic 2.0.1*)

The version number of the dictionary to which the current data block conforms.

[audit_conform]

AUDIT_CONTACT_AUTHOR

Data items in the AUDIT_CONTACT_AUTHOR category record details about the name and address of the author to be contacted concerning the content of this data block.

Category group(s): **inclusive_group**
audit_group
Category key(s): **_audit_contact_author.name**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_audit_contact_author.name   'Fitzgerald, Paula M.D.'
_audit_contact_author.address
; Department of Biophysical Chemistry
  Merck Research Laboratories
  PO Box 2000, Ry80M203
  Rahway, New Jersey 07065
  USA
;
_audit_contact_author.phone  '1(908)5945510'
_audit_contact_author.fax    '1(908)5946645'
_audit_contact_author.email  'paula_fitzgerald@merck.com'
```

_audit_contact_author.address (*text*)
_audit_contact_author_address (*cif_core.dic 2.0.1*)

The mailing address of the author of the data block to whom correspondence should be addressed.

Example:
```
; Department
  Institute
  Street
  City and postcode
  COUNTRY
;
```
[audit_contact_author]

_audit_contact_author.email (*line*)
_audit_contact_author_email (*cif_core.dic 2.0.1*)

The electronic mail address of the author of the data block to whom correspondence should be addressed, in a form recognizable to international networks. The format of e-mail addresses is given in Section 3.4, *Address Specification*, of *Internet Message Format*, RFC 2822, P. Resnick (Editor), Network Standards Group, April 2001.

Examples: 'name@host.domain.country', 'bm@iucr.org'.
[audit_contact_author]

_audit_contact_author.fax (*line*)
_audit_contact_author_fax (*cif_core.dic 2.0.1*)

The facsimile telephone number of the author of the data block to whom correspondence should be addressed. The recommended style starts with the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces.

Examples: '12(34)9477334', '12()349477334'. **[audit_contact_author]**

***_audit_contact_author.name** (*line*)
_audit_contact_author_name (*cif_core.dic 2.0.1*)

The name of the author of the data block to whom correspondence should be addressed. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.'. **[audit_contact_author]**

_audit_contact_author.phone (*line*)
_audit_contact_author_phone (*cif_core.dic 2.0.1*)

The telephone number of the author of the data block to whom correspondence should be addressed. The recommended style starts with the international dialing prefix, followed by the area code in parentheses, followed by the local number and any extension number prefixed by 'x', with no spaces.

Examples: '12(34)9477330', '12()349477330', '12(34)9477330x5543'.
[audit_contact_author]

AUDIT_LINK

Data items in the AUDIT_LINK category record details about the relationships between data blocks in the current CIF.

Category key(s): `_audit_link.block_code`
`_audit_link.block_description`

Example 1 – multiple structure paper, as illustrated in A Guide to CIF for Authors (1995). IUCr: Chester.

```
loop_
_audit_link.block_code
_audit_link.block_description
   .         'discursive text of paper with two structures'
   morA_(1)  'structure 1 of 2'
   morA_(2)  'structure 2 of 2'
```

Example 2 – example file for the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
_audit_link.block_code
_audit_link.block_description
 .        'publication details'
 KSE_COM  'experimental data common to ref./mod. structures'
 KSE_REF  'reference structure'
 KSE_MOD  'modulated structure'
```

*_audit_link.block_code (*code*)

_audit_link_block_code (*cif_core.dic 2.3*)

The value of `_audit_block.code` associated with a data block in the current file related to the current data block. The special value '.' may be used to refer to the current data block for completeness.

[audit_link]

*_audit_link.block_description (*text*)

_audit_link_block_description (*cif_core.dic 2.3*)

A textual description of the relationship of the referenced data block to the current one.

[audit_link]

CELL

Data items in the CELL category record details about the crystallographic cell parameters.

Category group(s): `inclusive_group`
`cell_group`

Category key(s): `_cell.entry_id`

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_cell.entry_id                   '5HVP'
_cell.length_a                   58.39
_cell.length_a_esd                0.05
_cell.length_b                   86.70
_cell.length_b_esd                0.12
_cell.length_c                   46.27
_cell.length_c_esd                0.06
_cell.angle_alpha                90.00
_cell.angle_beta                 90.00
_cell.angle_gamma                90.00
_cell.volume                     234237
_cell.details
; The cell parameters were refined every twenty frames during
  data integration. The cell lengths given are the mean of
  55 such refinements; the esds given are the root mean
  square deviations of these 55 observations from that mean.
;
```

Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_cell.length_a                    5.959
_cell.length_a_esd                0.001
_cell.length_b                   14.956
_cell.length_b_esd                0.001
_cell.length_c                   19.737
_cell.length_c_esd                0.003
_cell.angle_alpha                90.0
_cell.angle_beta                 90.0
_cell.angle_gamma                90.0
_cell.volume                   1759.0
_cell.volume_esd                  0.3
```

`_cell.angle_alpha` (*float, su*)

_cell_angle_alpha (*cif_core.dic 2.0.1*)

Unit-cell angle α of the reported structure in degrees.

The permitted range is [0.0, 180.0].

Related item: `_cell.angle_alpha_esd` (associated esd). Where no value is given, the assumed value is '`90.0`'. **[cell]**

`_cell.angle_alpha_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_cell.angle_alpha`.

Related item: `_cell.angle_alpha` (associated value). **[cell]**

`_cell.angle_beta` (*float, su*)

_cell_angle_beta (*cif_core.dic 2.0.1*)

Unit-cell angle β of the reported structure in degrees.

The permitted range is [0.0, 180.0].

Related item: `_cell.angle_beta_esd` (associated esd). Where no value is given, the assumed value is '`90.0`'. **[cell]**

`_cell.angle_beta_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_cell.angle_beta`.

Related item: `_cell.angle_beta` (associated value). **[cell]**

`_cell.angle_gamma` (*float, su*)

_cell_angle_gamma (*cif_core.dic 2.0.1*)

Unit-cell angle γ of the reported structure in degrees.

The permitted range is [0.0, 180.0].

Related item: `_cell.angle_gamma_esd` (associated esd). Where no value is given, the assumed value is '`90.0`'. **[cell]**

`_cell.angle_gamma_esd` (*float*)

The standard uncertainty (estimated standard deviation) of `_cell.angle_gamma`.

Related item: `_cell.angle_gamma` (associated value). **[cell]**

`_cell.details` (*text*)

_cell_special_details (*cif_core.dic 2.0.1*)

A description of special aspects of the cell choice, noting possible alternative settings.

Examples: '`pseudo-orthorhombic`',
'`standard setting from 45 deg rotation around c`'. **[cell]**

*`_cell.entry_id`

This data item is a pointer to `_entry.id` in the ENTRY category.

`_cell.formula_units_Z` (*int*)

_cell_formula_units_Z (*cif_core.dic 2.0.1*)

The number of the formula units in the unit cell as specified by `_chemical_formula.structural`, `_chemical_formula.moiety` or `_chemical_formula.sum`.

The permitted range is $[1, \infty)$. **[cell]**

_cell.length_a (*float, su*)
_cell_length_a (*cif_core.dic 2.0.1*)
Unit-cell length a corresponding to the structure reported in ångströms.
The permitted range is $[0.0, \infty)$.
Related item: **_cell.length_a_esd** (associated esd). **[cell]**

_cell.length_a_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_cell.length_a**.
Related item: **_cell.length_a** (associated value). **[cell]**

_cell.length_b (*float, su*)
_cell_length_b (*cif_core.dic 2.0.1*)
Unit-cell length b corresponding to the structure reported in ångströms.
The permitted range is $[0.0, \infty)$.
Related item: **_cell.length_b_esd** (associated esd). **[cell]**

_cell.length_b_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_cell.length_b**.
Related item: **_cell.length_b** (associated value). **[cell]**

_cell.length_c (*float, su*)
_cell_length_c (*cif_core.dic 2.0.1*)
Unit-cell length c corresponding to the structure reported in ångströms.
The permitted range is $[0.0, \infty)$.
Related item: **_cell.length_c_esd** (associated esd). **[cell]**

_cell.length_c_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_cell.length_c**.
Related item: **_cell.length_c** (associated value). **[cell]**

_cell.reciprocal_angle_alpha (*float, su*)
_cell_reciprocal_angle_alpha (*cif_core.dic 2.3*)
The angle α^* defining the reciprocal cell in degrees. α^*, β^* and $\gamma*$ are related to the angles in the real cell by

$$\cos\alpha^* = (\cos\beta\cos\gamma - \cos\alpha)/(\sin\beta\sin\gamma),$$
$$\cos\beta^* = (\cos\gamma\cos\alpha - \cos\beta)/(\sin\gamma\sin\alpha),$$
$$\cos\gamma^* = (\cos\alpha\cos\beta - \cos\gamma)/(\sin\alpha\sin\beta).$$

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.
The permitted range is [0.0, 180.0].
Related item: **_cell.reciprocal_angle_alpha_esd** (associated esd). Where no value is given, the assumed value is '90.0'. **[cell]**

_cell.reciprocal_angle_alpha_esd (*float*)
The estimated standard deviation of **_cell.reciprocal_angle_alpha**.
Related item: **_cell.reciprocal_angle_alpha** (associated value). **[cell]**

_cell.reciprocal_angle_beta (*float, su*)
_cell_reciprocal_angle_beta (*cif_core.dic 2.3*)
The angle β^* defining the reciprocal cell in degrees. α^*, β^* and γ^* are related to the angles in the real cell by

$$\cos\alpha^* = (\cos\beta\cos\gamma - \cos\alpha)/(\sin\beta\sin\gamma),$$
$$\cos\beta^* = (\cos\gamma\cos\alpha - \cos\beta)/(\sin\gamma\sin\alpha),$$
$$\cos\gamma^* = (\cos\alpha\cos\beta - \cos\gamma)/(\sin\alpha\sin\beta).$$

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.
The permitted range is [0.0, 180.0].
Related item: **_cell.reciprocal_angle_beta_esd** (associated esd). Where no value is given, the assumed value is '90.0'. **[cell]**

_cell.reciprocal_angle_beta_esd (*float*)
The estimated standard deviation of **_cell.reciprocal_angle_beta**.
Related item: **_cell.reciprocal_angle_beta** (associated value). **[cell]**

_cell.reciprocal_angle_gamma (*float, su*)
_cell_reciprocal_angle_gamma (*cif_core.dic 2.3*)
The angle γ^* defining the reciprocal cell in degrees. α^*, β^* and γ^* are related to the angles in the real cell by

$$\cos\alpha^* = (\cos\beta\cos\gamma - \cos\alpha)/(\sin\beta\sin\gamma),$$
$$\cos\beta^* = (\cos\gamma\cos\alpha - \cos\beta)/(\sin\gamma\sin\alpha),$$
$$\cos\gamma^* = (\cos\alpha\cos\beta - \cos\gamma)/(\sin\alpha\sin\beta).$$

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.
The permitted range is [0.0, 180.0].
Related item: **_cell.reciprocal_angle_gamma_esd** (associated esd). Where no value is given, the assumed value is '90.0'. **[cell]**

_cell.reciprocal_angle_gamma_esd (*float*)
The estimated standard deviation of **_cell.reciprocal_angle_gamma**.
Related item: **_cell.reciprocal_angle_gamma** (associated value). **[cell]**

_cell.reciprocal_length_a (*float, su*)
_cell_reciprocal_length_a (*cif_core.dic 2.3*)
The reciprocal-cell length a^* in inverse ångströms. a^*, b^* and c^* are related to the lengths in the real cell by

$$a^* = bc\sin\alpha/V,$$
$$b^* = ca\sin\beta/V,$$
$$c^* = ab\sin\gamma/V,$$

where V is the cell volume.
Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.
The permitted range is $[0.0, \infty)$.
Related item: **_cell.reciprocal_length_a_esd** (associated esd). **[cell]**

_cell.reciprocal_length_a_esd (*float*)
The estimated standard deviation of **_cell.reciprocal_length_a**.
Related item: **_cell.reciprocal_length_a** (associated value). **[cell]**

_cell.reciprocal_length_b (*float, su*)
_cell_reciprocal_length_b (*cif_core.dic 2.3*)
The reciprocal-cell length b^* in inverse ångströms. a^*, b^* and c^* are related to the lengths in the real cell by

$$a^* = bc\sin\alpha/V,$$
$$b^* = ca\sin\beta/V,$$
$$c^* = ab\sin\gamma/V,$$

where V is the cell volume.

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.

The permitted range is $[0.0, \infty)$.

Related item: **_cell.reciprocal_length_b_esd** (associated esd). **[cell]**

_cell.reciprocal_length_b_esd (*float*)
The estimated standard deviation of **_cell.reciprocal_length_b**.

Related item: **_cell.reciprocal_length_b** (associated value). **[cell]**

_cell.reciprocal_length_c (*float, su*)
_cell_reciprocal_length_c (*cif_core.dic 2.3*)
The reciprocal-cell length c^* in inverse ångströms. a^*, b^* and c^* are related to the lengths in the real cell by

$$a^* = bc\sin\alpha/V,$$
$$b^* = ca\sin\beta/V,$$
$$c^* = ab\sin\gamma/V,$$

where V is the cell volume.

Reference: Buerger, M. J. (1942). *X-ray Crystallography*, p. 360. New York: John Wiley & Sons Inc.

The permitted range is $[0.0, \infty)$.

Related item: **_cell.reciprocal_length_c_esd** (associated esd). **[cell]**

_cell.reciprocal_length_c_esd (*float*)
The estimated standard deviation of **_cell.reciprocal_length_c**.

Related item: **_cell.reciprocal_length_c** (associated value). **[cell]**

_cell.volume (*float, su*)
_cell_volume (*cif_core.dic 2.0.1*)
Cell volume V in ångströms cubed.

$$V = abc(1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma)^{1/2},$$

where a = **_cell.length_a**, b = **_cell.length_b**, c = **_cell.length_c**, α = **_cell.angle_alpha**, β = **_cell.angle_beta** and γ = **_cell.angle_gamma**.

The permitted range is $[0.0, \infty)$.

Related item: **_cell.volume_esd** (associated esd). **[cell]**

_cell.volume_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_cell.volume**.

Related item: **_cell.volume** (associated value). **[cell]**

_cell.Z_PDB (*int*)
The number of the polymeric chains in a unit cell. In the case of heteropolymers, Z is the number of occurrences of the most populous chain. This data item is provided for compatibility with the original Protein Data Bank format, and only for that purpose.

The permitted range is $[1, \infty)$. **[cell]**

CELL_MEASUREMENT

Data items in the CELL_MEASUREMENT category record details about the measurement of the crystallographic cell parameters.

Category group(s): **inclusive_group**
cell_group
Category key(s): **_cell_measurement.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_cell_measurement.entry_id            '5HVP'
_cell_measurement.temp                293
_cell_measurement.temp_esd            3
_cell_measurement.theta_min           11
_cell_measurement.theta_max           31
_cell_measurement.wavelength          1.54
```

Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_cell_measurement.temp                293
_cell_measurement.reflns_used         25
_cell_measurement.theta_min           25
_cell_measurement.theta_max           31
```

***_cell_measurement.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_cell_measurement.pressure (*float, su*)
_cell_measurement_pressure (*cif_core.dic 2.0.1*)
The pressure in kilopascals at which the unit-cell parameters were measured (not the pressure at which the sample was synthesized).

Related item: **_cell_measurement.pressure_esd** (associated esd).
[cell_measurement]

_cell_measurement.pressure_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_cell_measurement.pressure**.

Related item: **_cell_measurement.pressure** (associated value).
[cell_measurement]

_cell_measurement.radiation (*line*)
_cell_measurement_radiation (*cif_core.dic 2.0.1*)
Description of the radiation used to measure the unit-cell data. See also **_cell_measurement.wavelength**.

Examples: 'neutron', 'Cu K\a', 'synchrotron'. **[cell_measurement]**

_cell_measurement.reflns_used (*int*)
_cell_measurement_reflns_used (*cif_core.dic 2.0.1*)
The total number of reflections used to determine the unit cell. These reflections may be specified as CELL_MEASUREMENT_REFLN data items.

[cell_measurement]

_cell_measurement.temp (*float, su*)
_cell_measurement_temperature (*cif_core.dic 2.0.1*)
The temperature in kelvins at which the unit-cell parameters were measured (not the temperature of synthesis).

The permitted range is $[0.0, \infty)$.

Related item: **_cell_measurement.temp_esd** (associated esd).
[cell_measurement]

_cell_measurement.temp_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_cell_measurement.temp**.

Related item: **_cell_measurement.temp** (associated value). **[cell_measurement]**

_cell_measurement.theta_max (*float*)
_cell_measurement_theta_max (*cif_core.dic 2.0.1*)
The maximum θ angle of reflections used to measure the unit cell in degrees.
The permitted range is [0.0, 90.0]. **[cell_measurement]**

_cell_measurement.theta_min (*float*)
_cell_measurement_theta_min (*cif_core.dic 2.0.1*)
The minimum θ angle of reflections used to measure the unit cell in degrees.
The permitted range is [0.0, 90.0]. **[cell_measurement]**

_cell_measurement.wavelength (*float*)
_cell_measurement_wavelength (*cif_core.dic 2.0.1*)
The wavelength in ångströms of the radiation used to measure the unit cell. If this is not specified, the wavelength is assumed to be that specified in the category DIFFRN_RADIATION_WAVELENGTH.
The permitted range is $[0.0, \infty)$. **[cell_measurement]**

CELL_MEASUREMENT_REFLN

Data items in the CELL_MEASUREMENT_REFLN category record details about the reflections used to determine the crystallographic cell parameters. The CELL_MEASUREMENT_REFLN data items would in general be used only for diffractometer data.
Category group(s): **inclusive_group**
cell_group
Category key(s): **_cell_measurement_refln.index_h**
_cell_measurement_refln.index_k
_cell_measurement_refln.index_l

Example 1 – extracted from the CAD-4 listing of $Rb_2S_2O_6$ at room temperature (unpublished).

```
loop_
_cell_measurement_refln.index_h
_cell_measurement_refln.index_k
_cell_measurement_refln.index_l
_cell_measurement_refln.theta
   -2    4    1          8.67
    0    3    2          9.45
    3    0    2          9.46
   -3    4    1          8.93
   -2    1   -2          7.53
   10    0    0         23.77
    0   10    0         23.78
   -5    4    1         11.14
  # - - - - data truncated for brevity - - - -
```

***_cell_measurement_refln.index_h** (*int*)
_cell_measurement_refln_index_h (*cif_core.dic 2.0.1*)
Miller index *h* of a reflection used for measurement of the unit cell.
[cell_measurement_refln]

***_cell_measurement_refln.index_k** (*int*)
_cell_measurement_refln_index_k (*cif_core.dic 2.0.1*)
Miller index *k* of a reflection used for measurement of the unit cell.
[cell_measurement_refln]

***_cell_measurement_refln.index_l** (*int*)
_cell_measurement_refln_index_l (*cif_core.dic 2.0.1*)
Miller index *l* of a reflection used for measurement of the unit cell.
[cell_measurement_refln]

_cell_measurement_refln.theta (*float*)
_cell_measurement_refln_theta (*cif_core.dic 2.0.1*)
θ angle for a reflection used for measurement of the unit cell in degrees.
The permitted range is [0.0, 90.0]. **[cell_measurement_refln]**

CHEM_COMP

Data items in the CHEM_COMP category give details about each of the chemical components from which the relevant chemical structures can be constructed, such as name, mass or charge. The related categories CHEM_COMP_ATOM, CHEM_COMP_BOND, CHEM_COMP_ANGLE *etc.* describe the detailed geometry of these chemical components.
Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp.id
_chem_comp.model_source
_chem_comp.name
  phe  '1987 Protin/Prolsq Ideals file' phenylalanine
  val  '1987 Protin/Prolsq Ideals file' alanine
# - - - - data truncated for brevity - - - -
```

_chem_comp.formula (*text*)
The formula for the chemical component. Formulae are written according to the following rules: (1) Only recognized element symbols may be used. (2) Each element symbol is followed by a 'count' number. A count of '1' may be omitted. (3) A space or parenthesis must separate each cluster of (element symbol + count), but in general parentheses are not used. (4) The order of elements depends on whether carbon is present or not. If carbon is present, the order should be: C, then H, then the other elements in alphabetical order of their symbol. If carbon is not present, the elements are listed purely in alphabetic order of their symbol. This is the 'Hill' system used by *Chemical Abstracts*.
Example: 'C18 H19 N7 O8 S'. **[chem_comp]**

_chem_comp.formula_weight (*float*)
Formula mass in daltons of the chemical component.
The permitted range is $[1.0, \infty)$. **[chem_comp]**

***_chem_comp.id** (*ucode*)
The value of **_chem_comp.id** must uniquely identify each item in the CHEM_COMP list. For protein polymer entities, this is the three-letter code for the amino acid. For nucleic acid polymer entities, this is the one-letter code for the base.
The following item(s) have an equivalent role in their respective categories:
_atom_site.label_comp_id,
_chem_comp.mon_nstd_parent_comp_id,
_chem_comp_atom.comp_id,
_chem_comp_chir.comp_id,
_chem_comp_chir_atom.comp_id,
_chem_comp_plane.comp_id,
_chem_comp_plane_atom.comp_id,
_entity_poly_seq.mon_id,
_chem_comp_angle.comp_id,
_chem_comp_bond.comp_id,
_chem_comp_tor.comp_id,
_chem_comp_tor_value.comp_id,
_geom_angle.atom_site_label_comp_id_1,
_geom_angle.atom_site_label_comp_id_2,
_geom_angle.atom_site_label_comp_id_3,
_geom_bond.atom_site_label_comp_id_1,
_geom_bond.atom_site_label_comp_id_2,
_geom_contact.atom_site_label_comp_id_1,
_geom_contact.atom_site_label_comp_id_2,
_geom_hbond.atom_site_label_comp_id_A,
_geom_hbond.atom_site_label_comp_id_D,

_geom_hbond.atom_site_label_comp_id_H,
_geom_torsion.atom_site_label_comp_id_1,
_geom_torsion.atom_site_label_comp_id_2,
_geom_torsion.atom_site_label_comp_id_3,
_geom_torsion.atom_site_label_comp_id_4,
_struct_conf.beg_label_comp_id,
_struct_conf.end_label_comp_id,
_struct_conn.ptnr1_label_comp_id,
_struct_conn.ptnr2_label_comp_id,
_struct_mon_nucl.label_comp_id,
_struct_mon_prot.label_comp_id,
_struct_mon_prot_cis.label_comp_id,
_struct_ncs_dom_lim.beg_label_comp_id,
_struct_ncs_dom_lim.end_label_comp_id,
_struct_ref_seq_dif.db_mon_id,
_struct_ref_seq_dif.mon_id,
_struct_sheet_range.beg_label_comp_id,
_struct_sheet_range.end_label_comp_id,
_struct_site_gen.label_comp_id.

Examples: 'ala', 'val', 'A', 'C'. **[chem_comp]**

_chem_comp.model_details (*text*)

A description of special aspects of the generation of the coordinates for the model of the component.

Example: 'geometry idealized but not minimized'. **[chem_comp]**

_chem_comp.model_erf (*line*)

A pointer to an external reference file from which the atomic description of the component is taken.

[chem_comp]

_chem_comp.model_source (*text*)

The source of the coordinates for the model of the component.

Examples: 'CSD entry ABCDEF', 'built using Quanta/Charmm'.

[chem_comp]

_chem_comp.mon_nstd_class (*text*)

A description of the class of a nonstandard monomer if the nonstandard monomer represents a modification of a standard monomer.

Examples: 'iodinated base', 'phosphorylated amino acid', 'brominated base', 'modified amino acid', 'glycosylated amino acid'.

[chem_comp]

_chem_comp.mon_nstd_details (*text*)

A description of special details of a nonstandard monomer.

[chem_comp]

_chem_comp.mon_nstd_flag (*ucode*)

'yes' indicates that this is a 'standard' monomer, 'no' indicates that it is 'nonstandard'. Nonstandard monomers should be described in more detail using the **_chem_comp.mon_nstd_parent**, **_chem_comp.mon_nstd_class** and **_chem_comp.mon_nstd_details** data items.

The data value must be one of the following:

no	the monomer is nonstandard
n	abbreviation for 'no'
yes	the monomer is standard
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[chem_comp]**

_chem_comp.mon_nstd_parent (*code*)

The name of the parent monomer of the nonstandard monomer, if the nonstandard monomer represents a modification of a standard monomer.

Examples: 'tyrosine', 'cytosine'. **[chem_comp]**

_chem_comp.mon_nstd_parent_comp_id

The identifier for the parent component of the nonstandard component. This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_chem_comp.name (*line*)

The full name of the component.

Examples: 'alanine', 'valine', 'adenine', 'cytosine'. **[chem_comp]**

_chem_comp.number_atoms_all (*int*)

The total number of atoms in the component.

The permitted range is $[1, \infty)$. **[chem_comp]**

_chem_comp.number_atoms_nh (*int*)

The number of non-hydrogen atoms in the component.

The permitted range is $[1, \infty)$. **[chem_comp]**

_chem_comp.one_letter_code (*uchar1*)

For standard polymer components, the one-letter code for the component. If there is not a standard one-letter code for this component, or if this is a non-polymer component, the one-letter code should be given as 'X'. This code may be preceded by a '+' character to indicate that the component is a modification of a standard component.

Examples: 'A' (alanine or adenine), 'B' (ambiguous asparagine/aspartic acid), 'R' (arginine), 'N' (asparagine), 'D' (aspartic acid), 'C' (cysteine or cystine or cytosine), 'Q' (glutamine), 'E' (glutamic acid), 'Z' (ambiguous glutamine/glutamic acid), 'G' (glycine or guanine), 'H' (histidine), 'I' (isoleucine), 'L' (leucine), 'K' (lysine), 'M' (methionine), 'F' (phenylalanine), 'P' (proline), 'S' (serine), 'T' (threonine or thymine), 'W' (tryptophan), 'Y' (tyrosine), 'V' (valine), 'U' (uracil), 'O' (water), 'X' (other). **[chem_comp]**

_chem_comp.three_letter_code (*uchar3*)

For standard polymer components, the three-letter code for the component. If there is not a standard three-letter code for this component, or if this is a non-polymer component, the three-letter code should be given as 'UNK'. This code may be preceded by a '+' character to indicate that the component is a modification of a standard component.

Examples: 'ALA' (alanine), 'ARG' (arginine), 'ASN' (asparagine), 'ASP' (aspartic acid), 'ASX' (ambiguous asparagine/aspartic acid), 'CYS' (cysteine), 'GLN' (glutamine), 'GLU' (glutamic acid), 'GLY' (glycine), 'GLX' (ambiguous glutamine/glutamic acid), 'HIS' (histidine), 'ILE' (isoleucine), 'LEU' (leucine), 'LYS' (lysine), 'MET' (methionine), 'PHE' (phenylalanine), 'PRO' (proline), 'SER' (serine), 'THR' (threonine), 'TRP' (tryptophan), 'TRY' (tyrosine), 'VAL' (valine), '1MA' (1-methyladenosine), '5MC' (5-methylcytosine), 'OMC' (2′-*O*-methylcytodine), '1MG' (1-methylguanosine), '2MG' (N(2)-methylguanosine), 'M2G' (N(2)-dimethylguanosine), '7MG' (7-methylguanosine), 'OMG' (2′-*O*-methylguanosine), 'H2U' (dihydrouridine), '5MU' (ribosylthymidine), 'PSU' (pseudouridine), 'ACE' (acetic acid), 'FOR' (formic acid), 'HOH' (water), 'UNK' (other). **[chem_comp]**

***_chem_comp.type** (*uline*)
For standard polymer components, the type of the monomer. Note that monomers that will form polymers are of three types: linking monomers, monomers with some type of N-terminal (or 5′) cap and monomers with some type of C-terminal (or 3′) cap.

The following item(s) have an equivalent role in their respective categories:

_chem_comp_link.type_comp_1,

_chem_comp_link.type_comp_2.

The data value must be one of the following:

```
'D-peptide linking'
'L-peptide linking'
'D-peptide NH3 amino terminus'
'L-peptide NH3 amino terminus'
'D-peptide COOH carboxy terminus'
'L-peptide COOH carboxy terminus'
'DNA linking'
'RNA linking'
'DNA OH 5 prime terminus'
'RNA OH 5 prime terminus'
'DNA OH 3 prime terminus'
'RNA OH 3 prime terminus'
'D-saccharide 1,4 and 1,4 linking'
'L-saccharide 1,4 and 1,4 linking'
'D-saccharide 1,4 and 1,6 linking'
'L-saccharide 1,4 and 1,6 linking'
L-saccharide
D-saccharide
saccharide
non-polymer
other
```
[chem_comp]

CHEM_COMP_ANGLE

Data items in the CHEM_COMP_ANGLE category record details about angles in a chemical component. Angles are designated by three atoms, with the second atom forming the vertex of the angle. Target values may be specified as angles in degrees, as a distance between the first and third atoms, or both.

Category group(s): **inclusive_group**
chem_comp_group

Category key(s): **_chem_comp_angle.comp_id**
_chem_comp_angle.atom_id_1
_chem_comp_angle.atom_id_2
_chem_comp_angle.atom_id_3

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_angle.comp_id
_chem_comp_angle.atom_id_1
_chem_comp_angle.atom_id_2
_chem_comp_angle.atom_id_3
_chem_comp_angle.value_angle
_chem_comp_angle.value_dist
  phe  N    CA   C    xxx.xx   x.xx
  phe  CA   C    O    xxx.xx   x.xx
  phe  CB   CA   C    xxx.xx   x.xx
  phe  CB   CA   N    xxx.xx   x.xx
  phe  CA   CB   CG   xxx.xx   x.xx
  phe  CB   CG   CD1  xxx.xx   x.xx
  phe  CB   CG   CD2  xxx.xx   x.xx
  phe  CD1  CG   CD2  xxx.xx   x.xx
  phe  CG   CD1  CE1  xxx.xx   x.xx
  phe  CD1  CE1  CZ   xxx.xx   x.xx
  phe  CE1  CZ   CE2  xxx.xx   x.xx
  phe  CZ   CE2  CD2  xxx.xx   x.xx
  phe  CG   CD2  CE2  xxx.xx   x.xx
  val  N    CA   C    xxx.xx   x.xx
  val  CA   C    O    xxx.xx   x.xx
  val  CB   CA   C    xxx.xx   x.xx
  val  CB   CA   N    xxx.xx   x.xx
  val  CA   CB   CG1  xxx.xx   x.xx
  val  CA   CB   CG2  xxx.xx   x.xx
  val  CG1  CB   CG2  xxx.xx   x.xx
```

***_chem_comp_angle.atom_id_1**
The ID of the first of the three atoms that define the angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_angle.atom_id_2**
The ID of the second of the three atoms that define the angle. The second atom is taken to be the apex of the angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_angle.atom_id_3**
The ID of the third of the three atoms that define the angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_angle.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_chem_comp_angle.value_angle (*float, su*)
The value that should be taken as the target value for the angle associated with the specified atoms, expressed in degrees.

The permitted range is [0.0, 180.0].

Related item: **_chem_comp_angle.value_angle_esd** (associated esd).

[chem_comp_angle]

_chem_comp_angle.value_angle_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_angle.value_angle**.

The permitted range is [0.0, 180.0].

Related item: **_chem_comp_angle.value_angle** (associated value).

[chem_comp_angle]

_chem_comp_angle.value_dist (*float, su*)
The value that should be taken as the target value for the angle associated with the specified atoms, expressed as the distance between the atoms specified by **_chem_comp_angle.atom_id_1** and **_chem_comp_angle.atom_id_3**.

The permitted range is [0.0, ∞).

Related item: **_chem_comp_angle.value_dist_esd** (associated esd).

[chem_comp_angle]

_chem_comp_angle.value_dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_angle.value_dist**.

The permitted range is [0.0, ∞).

Related item: **_chem_comp_angle.value_dist** (associated value).

[chem_comp_angle]

CHEM_COMP_ATOM

Data items in the CHEM_COMP_ATOM category record details about the atoms in a chemical component. Specifying the atomic coordinates for the components in this category is an alternative to specifying the structure of the component *via* bonds, angles, planes *etc.* in the appropriate CHEM_COMP subcategories.

Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_atom.comp_id**
_chem_comp_atom.atom_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_atom.comp_id
_chem_comp_atom.atom_id
_chem_comp_atom.type_symbol
_chem_comp_atom.substruct_code
_chem_comp_atom.model_Cartn_x
_chem_comp_atom.model_Cartn_y
_chem_comp_atom.model_Cartn_z
  phe  N    N  main   1.20134   0.84658   0.00000
  phe  CA   C  main   0.00000   0.00000   0.00000
  phe  C    C  main  -1.25029   0.88107   0.00000
  phe  O    O  main  -2.18525   0.66029  -0.78409
  phe  CB   C  side   0.00662  -1.03603   1.11081
  phe  CG   C  side   0.03254  -0.49711   2.50951
  phe  CD1  C  side  -1.15813  -0.12084   3.13467
  phe  CE1  C  side  -1.15720   0.38038   4.42732
  phe  CZ   C  side   0.05385   0.51332   5.11032
  phe  CE2  C  side   1.26137   0.11613   4.50975
  phe  CD2  C  side   1.23668  -0.38351   3.20288
  val  N    N  main   1.20134   0.84658   0.00000
  val  CA   C  main   0.00000   0.00000   0.00000
  val  C    C  main  -1.25029   0.88107   0.00000
  val  O    O  main  -2.18525   0.66029  -0.78409
  val  CB   C  side   0.05260  -0.99339   1.17429
  val  CG1  C  side  -0.13288  -0.31545   2.52668
  val  CG2  C  side  -0.94265  -2.12930   0.99811
```

_chem_comp_atom.alt_atom_id (*line*)

An alternative identifier for the atom. This data item would be used in cases where alternative nomenclatures exist for labelling atoms in a group.

[chem_comp_atom]

*_chem_comp_atom.atom_id (*atcode*)

The value of **_chem_comp_atom.atom_id** must uniquely identify each atom in each monomer in the CHEM_COMP_ATOM list. The atom identifiers need not be unique over all atoms in the data block; they need only be unique for each atom in a component. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_atom_site.label_atom_id,
_chem_comp_angle.atom_id_1,
_chem_comp_angle.atom_id_2,
_chem_comp_angle.atom_id_3,
_chem_comp_bond.atom_id_1,
_chem_comp_bond.atom_id_2,
_chem_comp_chir.atom_id,
_chem_comp_chir_atom.atom_id,
_chem_comp_plane_atom.atom_id,
_chem_comp_tor.atom_id_1,
_chem_comp_tor.atom_id_2,
_chem_comp_tor.atom_id_3,
_chem_comp_tor.atom_id_4,
_geom_angle.atom_site_label_atom_id_1,
_geom_angle.atom_site_label_atom_id_2,
_geom_angle.atom_site_label_atom_id_3,
_geom_bond.atom_site_label_atom_id_1,
_geom_bond.atom_site_label_atom_id_2,
_geom_contact.atom_site_label_atom_id_1,
_geom_contact.atom_site_label_atom_id_2,
_geom_hbond.atom_site_label_atom_id_A,
_geom_hbond.atom_site_label_atom_id_D,
_geom_hbond.atom_site_label_atom_id_H,
_geom_torsion.atom_site_label_atom_id_1,
_geom_torsion.atom_site_label_atom_id_2,
_geom_torsion.atom_site_label_atom_id_3,
_geom_torsion.atom_site_label_atom_id_4,
_struct_conn.ptnr1_label_atom_id,
_struct_conn.ptnr2_label_atom_id,
_struct_sheet_hbond.range_1_beg_label_atom_id,
_struct_sheet_hbond.range_1_end_label_atom_id,
_struct_sheet_hbond.range_2_beg_label_atom_id,
_struct_sheet_hbond.range_2_end_label_atom_id,
_struct_site_gen.label_atom_id. **[chem_comp_atom]**

_chem_comp_atom.charge (*int*)

The net integer charge assigned to this atom. This is the formal charge assignment normally found in chemical diagrams.

The permitted range is $[-8, 8]$. Where no value is given, the assumed value is '0'.

Examples: '1' (for an ammonium nitrogen), '-1' (for a chloride ion).

[chem_comp_atom]

*_chem_comp_atom.comp_id

This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_chem_comp_atom.model_Cartn_x (*float, su*)

The *x* component of the coordinates for this atom in this component specified as orthogonal ångströms. The choice of reference axis frame for the coordinates is arbitrary. The set of coordinates input for the entity here is intended to correspond to the atomic model used to generate restraints for structure refinement, not to atom sites in the ATOM_SITE list.

Related item: **_chem_comp_atom.model_Cartn_x_esd** (associated esd).

[chem_comp_atom]

_chem_comp_atom.model_Cartn_x_esd (*float*)

The standard uncertainty (estimated standard deviation) of **_chem_comp_atom.model_Cartn_x**.

Related item: **_chem_comp_atom.model_Cartn_x** (associated value).

[chem_comp_atom]

_chem_comp_atom.model_Cartn_y (*float, su*)

The *y* component of the coordinates for this atom in this component specified as orthogonal ångströms. The choice of reference axis frame for the coordinates is arbitrary. The set of coordinates input for the entity here is intended to correspond to the atomic model used to generate restraints for structure refinement, not to atom sites in the ATOM_SITE list.

Related item: **_chem_comp_atom.model_Cartn_y_esd** (associated esd).

[chem_comp_atom]

_chem_comp_atom.model_Cartn_y_esd (*float*)

The standard uncertainty (estimated standard deviation) of **_chem_comp_atom.model_Cartn_y**.

Related item: **_chem_comp_atom.model_Cartn_y** (associated value).

[chem_comp_atom]

_chem_comp_atom.model_Cartn_z (*float, su*)
The z component of the coordinates for this atom in this component specified as orthogonal ångströms. The choice of reference axis frame for the coordinates is arbitrary. The set of coordinates input for the entity here is intended to correspond to the atomic model used to generate restraints for structure refinement, not to atom sites in the ATOM_SITE list.
Related item: **_chem_comp_atom.model_Cartn_z_esd** (associated esd).
[chem_comp_atom]

_chem_comp_atom.model_Cartn_z_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_atom.model_Cartn_z**.
Related item: **_chem_comp_atom.model_Cartn_z** (associated value).
[chem_comp_atom]

_chem_comp_atom.partial_charge (*float*)
The partial charge assigned to this atom.
[chem_comp_atom]

_chem_comp_atom.substruct_code (*ucode*)
This data item assigns the atom to a substructure of the component, if appropriate.
The data value must be one of the following:

main	main chain of an amino acid
side	side chain of an amino acid
base	base of a nucleic acid
phos	phosphate of a nucleic acid
sugar	sugar of a nucleic acid
none	not appropriate for this monomer

[chem_comp_atom]

***_chem_comp_atom.type_symbol**
This data item is a pointer to **_atom_type.symbol** in the ATOM_TYPE category.

CHEM_COMP_BOND

Data items in the CHEM_COMP_BOND category record details about the bonds between atoms in a chemical component. Target values may be specified as bond orders, as a distance between the two atoms, or both.
Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_bond.comp_id**
_chem_comp_bond.atom_id_1
_chem_comp_bond.atom_id_2

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_bond.comp_id
_chem_comp_bond.atom_id_1
_chem_comp_bond.atom_id_2
_chem_comp_bond.value_order
  phe  N    CA   sing
  phe  CA   C    sing
  phe  C    O    doub
  phe  CB   CA   sing
  phe  CB   CG   sing
  phe  CG   CD1  arom
  phe  CD1  CE1  arom
  phe  CE1  CZ   arom
  phe  CZ   CE2  arom
  phe  CE2  CD2  arom
  phe  CD2  CG   arom
  val  N    CA   sing
  val  CA   C    sing
  val  C    O    doub
  val  CB   CA   sing
  val  CB   CG1  sing
  val  CB   CG2  sing
```

***_chem_comp_bond.atom_id_1**
The ID of the first of the two atoms that define the bond. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_bond.atom_id_2**
The ID of the second of the two atoms that define the bond. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_bond.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_chem_comp_bond.value_dist (*float, su*)
The value that should be taken as the target for the chemical bond associated with the specified atoms, expressed as a distance.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_comp_bond.value_dist_esd** (associated esd).
[chem_comp_bond]

_chem_comp_bond.value_dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_bond.value_dist**.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_comp_bond.value_dist** (associated value).
[chem_comp_bond]

_chem_comp_bond.value_order (*ucode*)
The value that should be taken as the target for the chemical bond associated with the specified atoms, expressed as a bond order.
The data value must be one of the following:

sing	single bond
doub	double bond
trip	triple bond
quad	quadruple bond
arom	aromatic bond
poly	polymeric bond
delo	delocalized double bond
pi	π bond

Where no value is given, the assumed value is 'sing'. **[chem_comp_bond]**

CHEM_COMP_CHIR

Data items in the CHEM_COMP_CHIR category provide details about the chiral centres in a chemical component. The atoms bonded to the chiral atom are specified in the CHEM_COMP_CHIR_ATOM category.
Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_chir.comp_id**
_chem_comp_chir.id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_chir.comp_id
_chem_comp_chir.id
_chem_comp_chir.atom_id
  phe  phe1  CA
  val  val1  CA
# - - - - data truncated for brevity - - - -
```

_chem_comp_chir.atom_config (*ucode*)
The chiral configuration of the atom that is a chiral centre.
The data value must be one of the following:

R	absolute configuration *R*
S	absolute configuration *S*

[chem_comp_chir]

***_chem_comp_chir.atom_id**
The ID of the atom that is a chiral centre. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_chir.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

***_chem_comp_chir.id** (*code*)
The value of **_chem_comp_chir.id** must uniquely identify a record in the CHEM_COMP_CHIR list.
The following item(s) have an equivalent role in their respective categories:
_chem_comp_chir_atom.chir_id. **[chem_comp_chir]**

_chem_comp_chir.number_atoms_all (*int*)
The total number of atoms bonded to the atom specified by **_chem_comp_chir.atom_id**.

[chem_comp_chir]

_chem_comp_chir.number_atoms_nh (*int*)
The number of non-hydrogen atoms bonded to the atom specified by **_chem_comp_chir.atom_id**.

[chem_comp_chir]

_chem_comp_chir.volume_flag (*ucode*)
A flag to indicate whether a chiral volume should match the standard value in both magnitude and sign, or in magnitude only.
The data value must be one of the following:

sign	match magnitude and sign
nosign	match magnitude only

[chem_comp_chir]

_chem_comp_chir.volume_three (*float, su*)
The chiral volume, V_c, for chiral centres that involve a chiral atom bonded to three non-hydrogen atoms and one hydrogen atom.

$$V_c = \mathbf{V}_1 \cdot (\mathbf{V}_2 \times \mathbf{V}_3),$$

where $\mathbf{V}_1$ = the vector distance from the atom specified by **_chem_comp_chir.atom_id** to the first atom in the CHEM_COMP_CHIR_ATOM list, $\mathbf{V}_2$ = the vector distance from the atom specified by **_chem_comp_chir.atom_id** to the second atom in the CHEM_COMP_CHIR_ATOM list, $\mathbf{V}_3$ = the vector distance from the atom specified by **_chem_comp_chir.atom_id** to the third atom in the CHEM_COMP_CHIR_ATOM list, $\cdot$ = the vector dot product and $\times$ = the vector cross product.
Related item: **_chem_comp_chir.volume_three_esd** (associated esd).

[chem_comp_chir]

_chem_comp_chir.volume_three_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_chir.volume_three**.
Related item: **_chem_comp_chir.volume_three** (associated value).

[chem_comp_chir]

CHEM_COMP_CHIR_ATOM

Data items in the CHEM_COMP_CHIR_ATOM category enumerate the atoms bonded to a chiral atom within a chemical component.
Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_chir_atom.chir_id**
_chem_comp_chir_atom.atom_id
_chem_comp_chir_atom.comp_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_chir_atom.comp_id
_chem_comp_chir_atom.chir_id
_chem_comp_chir_atom.atom_id
  phe  1  N
  phe  1  C
  phe  1  CB
  val  1  N
  val  1  C
  val  1  CB
   # - - - - data truncated for brevity - - - -
```

***_chem_comp_chir_atom.atom_id**
The ID of an atom bonded to the chiral atom.
This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_chem_comp_chir_atom.chir_id**
This data item is a pointer to **_chem_comp_chir.id** in the CHEM_COMP_CHIR category.

***_chem_comp_chir_atom.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_chem_comp_chir_atom.dev (*float*)
The standard uncertainty (estimated standard deviation) of the position of this atom from the plane defined by all of the atoms in the plane.

[chem_comp_chir_atom]

CHEM_COMP_LINK

Data items in the CHEM_COMP_LINK category give details about the links between chemical components.
Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_comp_link.link_id**

_chem_comp_link.details (*text*)
A description of special aspects of a link between chemical components in the structure.

[chem_comp_link]

***_chem_comp_link.link_id**
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

***_chem_comp_link.type_comp_1**
The type of the first of the two components joined by the link. This data item is a pointer to **_chem_comp.type** in the CHEM_COMP category.

* **_chem_comp_link.type_comp_2**
The type of the second of the two components joined by the link. This data item is a pointer to **_chem_comp.type** in the CHEM_COMP category.

CHEM_COMP_PLANE

Data items in the CHEM_COMP_PLANE category provide identifiers for the planes in a chemical component. The atoms in the plane are specified in the CHEM_COMP_PLANE_ATOM category.

Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_plane.comp_id**
_chem_comp_plane.id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_plane.comp_id
_chem_comp_plane.id
  phe  phe1
```

* **_chem_comp_plane.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

* **_chem_comp_plane.id** (*code*)
The value of **_chem_comp_plane.id** must uniquely identify a record in the CHEM_COMP_PLANE list.

The following item(s) have an equivalent role in their respective categories:

_chem_comp_plane_atom.plane_id. **[chem_comp_plane]**

_chem_comp_plane.number_atoms_all (*int*)
The total number of atoms in the plane.

[chem_comp_plane]

_chem_comp_plane.number_atoms_nh (*int*)
The number of non-hydrogen atoms in the plane.

[chem_comp_plane]

CHEM_COMP_PLANE_ATOM

Data items in the CHEM_COMP_PLANE_ATOM category enumerate the atoms in a plane within a chemical component.

Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_plane_atom.plane_id**
_chem_comp_plane_atom.atom_id
_chem_comp_plane_atom.comp_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_plane_atom.plane_id
_chem_comp_plane_atom.comp_id
_chem_comp_plane_atom.atom_id
  phe1  phe  CB
  phe1  phe  CG
  phe1  phe  CD1
  phe1  phe  CE1
  phe1  phe  CZ
  phe1  phe  CE2
  phe1  phe  CD2
```

* **_chem_comp_plane_atom.atom_id**
The ID of an atom involved in the plane. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

* **_chem_comp_plane_atom.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_chem_comp_plane_atom.dist_esd (*float*)
This data item is the standard deviation of the out-of-plane distance for this atom.

* **_chem_comp_plane_atom.plane_id**
This data item is a pointer to **_chem_comp_plane.id** in the CHEM_COMP_PLANE category.

CHEM_COMP_TOR

Data items in the CHEM_COMP_TOR category record details about the torsion angles in a chemical component. As torsion angles can have more than one target value, the target values are specified in the CHEM_COMP_TOR_VALUE category.

Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_tor.comp_id**
_chem_comp_tor.id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_tor.comp_id
_chem_comp_tor.id
_chem_comp_tor.atom_id_1
_chem_comp_tor.atom_id_2
_chem_comp_tor.atom_id_3
_chem_comp_tor.atom_id_4
  phe  phe_chi1   N    CA   CB   CG
  phe  phe_chi2   CA   CB   CG   CD1
  phe  phe_ring1  CB   CG   CD1  CE1
  phe  phe_ring2  CB   CG   CD2  CE2
  phe  phe_ring3  CG   CD1  CE1  CZ
  phe  phe_ring4  CD1  CE1  CZ   CE2
  phe  phe_ring5  CE1  CZ   CE2  CD2
```

* **_chem_comp_tor.atom_id_1**
The ID of the first of the four atoms that define the torsion angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

* **_chem_comp_tor.atom_id_2**
The ID of the second of the four atoms that define the torsion angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

* **_chem_comp_tor.atom_id_3**
The ID of the third of the four atoms that define the torsion angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

* **_chem_comp_tor.atom_id_4**
The ID of the fourth of the four atoms that define the torsion angle. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

* **_chem_comp_tor.comp_id**
This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

***_chem_comp_tor.id** (*code*)
The value of **_chem_comp_tor.id** must uniquely identify a record in the CHEM_COMP_TOR list.
The following item(s) have an equivalent role in their respective categories:
_chem_comp_tor_value.tor_id. **[chem_comp_tor]**

CHEM_COMP_TOR_VALUE

Data items in the CHEM_COMP_TOR_VALUE category record details about the target values for the torsion angles enumerated in the CHEM_COMP_TOR list. Target values may be specified as angles in degrees, as a distance between the first and fourth atoms, or both.
Category group(s): **inclusive_group**
chem_comp_group
Category key(s): **_chem_comp_tor_value.tor_id**
_chem_comp_tor_value.comp_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_chem_comp_tor_value.tor_id
_chem_comp_tor_value.comp_id
_chem_comp_tor_value.angle
_chem_comp_tor_value.dist
  phe_chi1   phe  -60.0  2.88
  phe_chi1   phe  180.0  3.72
  phe_chi1   phe   60.0  2.88
  phe_chi2   phe   90.0  3.34
  phe_chi2   phe  -90.0  3.34
  phe_ring1  phe  180.0  3.75
  phe_ring2  phe  180.0  3.75
  phe_ring3  phe    0.0  2.80
  phe_ring4  phe    0.0  2.80
  phe_ring5  phe    0.0  2.80
```

***_chem_comp_tor_value.angle** (*float, su*)
A value that should be taken as a potential target value for the torsion angle associated with the specified atoms, expressed in degrees.
The permitted range is $[-180.0, 180.0]$.
Related item: **_chem_comp_tor_value.angle_esd** (associated esd).
[chem_comp_tor_value]

***_chem_comp_tor_value.angle_esd** (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_tor_value.angle**.
The permitted range is $[-180.0, 180.0]$.
Related item: **_chem_comp_tor_value.angle** (associated value).
[chem_comp_tor_value]

***_chem_comp_tor_value.comp_id**
This data item is a pointer to **_chem_comp_atom.comp_id** in the CHEM_COMP_ATOM category.

_chem_comp_tor_value.dist (*float, su*)
A value that should be taken as a potential target value for the torsion angle associated with the specified atoms, expressed as the distance between the atoms specified by **_chem_comp_tor.atom_id_1** and **_chem_comp_tor.atom_id_4** in the referenced record in the CHEM_COMP_TOR list. Note that the torsion angle cannot be fully specified by a distance (for instance, a torsion angle of $-60°$ will yield the same distance as a $60°$ angle). However, the distance specification can be useful for refinement in situations in which the angle is already close to the desired value.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_comp_tor_value.dist_esd** (associated esd).
[chem_comp_tor_value]

_chem_comp_tor_value.dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_tor_value.dist**.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_comp_tor_value.dist** (associated value).
[chem_comp_tor_value]

***_chem_comp_tor_value.tor_id**
This data item is a pointer to **_chem_comp_tor.id** in the CHEM_COMP_TOR category.

CHEM_LINK

Data items in the CHEM_LINK category give details about the links between chemical components.
Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link.id**

_chem_link.details (*text*)
A description of special aspects of a link between chemical components in the structure.
[chem_link]

***_chem_link.id** (*code*)
The value of **_chem_link.id** must uniquely identify each item in the CHEM_LINK list.
The following item(s) have an equivalent role in their respective categories:
_chem_link_angle.link_id,
_chem_link_bond.link_id,
_chem_link_chir.link_id,
_chem_link_plane.link_id,
_chem_link_tor.link_id,
_chem_comp_link.link_id,
_entity_link.link_id.
Examples: ‘peptide’, ‘oligosaccharide 1,4’, ‘DNA’. **[chem_link]**

CHEM_LINK_ANGLE

Data items in the CHEM_LINK_ANGLE category record details about angles in a link between chemical components.
Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_angle.link_id**
_chem_link_angle.atom_id_1
_chem_link_angle.atom_id_2
_chem_link_angle.atom_id_3

*Example 1 – Engh & Huber parameters [Acta Cryst. (1991), A**47**, 392–400] as interpreted by J. P. Priestle (1995). Consistent Stereochemical Dictionaries for Refinement and Model Building. CCP4 Daresbury Study Weekend, DL-CONF-95-001, ISSN 1358-6254. Warrington: Daresbury Laboratory.*

```
loop_
_chem_link_angle.link_id
_chem_link_angle.value_angle
_chem_link_angle.value_angle_esd
_chem_link_angle.atom_id_1
_chem_link_angle.atom_1_comp_id
_chem_link_angle.atom_id_2
_chem_link_angle.atom_2_comp_id
_chem_link_angle.atom_id_3
_chem_link_angle.atom_3_comp_id
 PEPTIDE  111.2  2.8  N  1  CA 1  C  1
 PEPTIDE  120.8  1.7  CA 1  C  1  O  1
 PEPTIDE  116.2  2.0  CA 1  C  1  N  2
 PEPTIDE  123.0  1.6  O  1  C  1  N  2
 PEPTIDE  121.7  1.8  C  1  N  2  CA 2
```

_chem_link_angle.atom_1_comp_id (*ucode*)
This data item indicates whether atom 1 is found in the first or the second of the two components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_angle]

_chem_link_angle.atom_2_comp_id (*ucode*)
This data item indicates whether atom 2 is found in the first or the second of the two components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_angle]

_chem_link_angle.atom_3_comp_id (*ucode*)
This data item indicates whether atom 3 is found in the first or the second of the two components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_angle]

***_chem_link_angle.atom_id_1** (*code*)
The ID of the first of the three atoms that define the angle. An atom with this ID must exist in the component of the type specified by **_chem_comp_link.type_comp_1** (or **_chem_comp_link.type_comp_2**, where the appropriate data item is indicated by the value of **_chem_comp_angle.atom_1_comp_id**).
[chem_link_angle]

***_chem_link_angle.atom_id_2** (*code*)
The ID of the second of the three atoms that define the angle. The second atom is taken to be the apex of the angle. An atom with this ID must exist in the component of the type specified by **_chem_comp_link.type_comp_1** (or **_chem_comp_link.type_comp_2**, where the appropriate data item is indicated by the value of **_chem_comp_angle.atom_2_comp_id**).
[chem_link_angle]

***_chem_link_angle.atom_id_3** (*code*)
The ID of the third of the three atoms that define the angle. An atom with this ID must exist in the component of the type specified by **_chem_comp_link.type_comp_1** (or **_chem_comp_link.type_comp_2**, where the appropriate data item is indicated by the value of **_chem_comp_angle.atom_3_comp_id**).
[chem_link_angle]

***_chem_link_angle.link_id**
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

_chem_link_angle.value_angle (*float, su*)
The value that should be taken as the target value for the angle associated with the specified atoms, expressed in degrees.
The permitted range is [0.0, 180.0].
Related item: **_chem_link_angle.value_angle_esd** (associated esd).
[chem_link_angle]

_chem_link_angle.value_angle_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_link_angle.value_angle**.
The permitted range is [0.0, 180.0].
Related item: **_chem_link_angle.value_angle** (associated value).
[chem_link_angle]

_chem_link_angle.value_dist (*float, su*)
The value that should be taken as the target value for the angle associated with the specified atoms, expressed as the distance between the atoms specified by **_chem_comp_angle.atom_id_1** and **_chem_comp_angle.atom_id_3**.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_link_angle.value_dist_esd** (associated esd).
[chem_link_angle]

_chem_link_angle.value_dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_comp_angle.value_dist**.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_link_angle.value_dist** (associated value).
[chem_link_angle]

CHEM_LINK_BOND

Data items in the CHEM_LINK_BOND category record details about bonds in a link between components in the chemical structure.

Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_bond.link_id**
_chem_link_bond.atom_id_1
_chem_link_bond.atom_id_2

*Example 1 – Engh & Huber parameters [Acta Cryst. (1991), A**47**, 392–400] as interpreted by J. P. Priestle (1995). Consistent Stereochemical Dictionaries for Refinement and Model Building. CCP4 Daresbury Study Weekend, DL-CONF-95-001, ISSN 1358-6254. Warrington: Daresbury Laboratory.*

```
loop_
_chem_link_bond.link_id
_chem_link_bond.value_dist
_chem_link_bond.value_dist_esd
_chem_link_bond.atom_id_1
_chem_link_bond.atom_1_comp_id
_chem_link_bond.atom_id_2
_chem_link_bond.atom_2_comp_id
 PEPTIDE  1.458  0.019  N  1  CA 1
 PEPTIDE  1.525  0.021  CA 1  C  1
 PEPTIDE  1.329  0.014  C  1  N  2
 PEPTIDE  1.231  0.020  C  1  O  1
```

_chem_link_bond.atom_1_comp_id (*ucode*)
This data item indicates whether atom 1 is found in the first or the second of the two components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_bond]

_chem_link_bond.atom_2_comp_id (*ucode*)
This data item indicates whether atom 2 is found in the first or the second of the two chemical components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_bond]

***_chem_link_bond.atom_id_1** (*code*)
The ID of the first of the two atoms that define the bond. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.
[chem_link_bond]

***_chem_link_bond.atom_id_2** (*code*)
The ID of the second of the two atoms that define the bond. As this data item does not point to a specific atom in a specific component, it is not a child in the linkage sense.
[chem_link_bond]

***_chem_link_bond.link_id**
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

_chem_link_bond.value_dist (*float, su*)
The value that should be taken as the target for the chemical bond associated with the specified atoms, expressed as a distance.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_link_bond.value_dist_esd** (associated esd).
[chem_link_bond]

_chem_link_bond.value_dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_link_bond.value_dist**.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_link_bond.value_dist** (associated value).
[chem_link_bond]

_chem_link_bond.value_order (*ucode*)
The value that should be taken as the target for the chemical bond associated with the specified atoms, expressed as a bond order.
The data value must be one of the following:

sing	single bond
doub	double bond
trip	triple bond
quad	quadruple bond
arom	aromatic bond
poly	polymeric bond
delo	delocalized double bond
pi	π bond

Where no value is given, the assumed value is '`sing`'. **[chem_link_bond]**

CHEM_LINK_CHIR

Data items in the CHEM_LINK_CHIR category provide details about the chiral centres in a link between two chemical components. The atoms bonded to the chiral atom are specified in the CHEM_LINK_CHIR_ATOM category.
Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_chir.link_id**
_chem_link_chir.id

_chem_link_chir.atom_comp_id (*ucode*)
This data item indicates whether the chiral atom is found in the first or the second of the two components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_chir]

_chem_link_chir.atom_config (*ucode*)
The chiral configuration of the atom that is a chiral centre.
The data value must be one of the following:

R	absolute configuration *R*
S	absolute configuration *S*

[chem_link_chir]

***_chem_link_chir.atom_id** (*code*)
The ID of the atom that is a chiral centre. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

***_chem_link_chir.id** (*code*)
The value of **_chem_link_chir.id** must uniquely identify a record in the CHEM_LINK_CHIR list.
The following item(s) have an equivalent role in their respective categories:
_chem_link_chir_atom.chir_id. **[chem_link_chir]**

***_chem_link_chir.link_id**
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

_chem_link_chir.number_atoms_all (*int*)
The total number of atoms bonded to the atom specified by **_chem_link_chir.atom_id**.
[chem_link_chir]

_chem_link_chir.number_atoms_nh (*int*)
The number of non-hydrogen atoms bonded to the atom specified by **_chem_link_chir.atom_id**.
[chem_link_chir]

_chem_link_chir.volume_flag (*ucode*)
A flag to indicate whether a chiral volume should match the standard value in both magnitude and sign, or in magnitude only.
The data value must be one of the following:

sign	match magnitude and sign
nosign	match magnitude only

[chem_link_chir]

_chem_link_chir.volume_three (*float, su*)
The chiral volume, V_c, for chiral centres that involve a chiral atom bonded to three non-hydrogen atoms and one hydrogen atom.

$$V_c = \mathbf{V}_1 \cdot (\mathbf{V}_2 \times \mathbf{V}_3),$$

where $\mathbf{V}_1$ = the vector distance from the atom specified by **_chem_link_chir.atom_id** to the first atom in the CHEM_LINK_CHIR_ATOM list, $\mathbf{V}_2$ = the vector distance from the atom specified by **_chem_link_chir.atom_id** to the second atom in the CHEM_LINK_CHIR_ATOM list, $\mathbf{V}_3$ = the vector distance from the atom specified by **_chem_link_chir.atom_id** to the third atom in the CHEM_LINK_CHIR_ATOM list, $\cdot$ = the vector dot product and $\times$ = the vector cross product.
Related item: **_chem_link_chir.volume_three_esd** (associated esd).
[chem_link_chir]

_chem_link_chir.volume_three_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_link_chir.volume_three**.
Related item: **_chem_link_chir.volume_three** (associated value).
[chem_link_chir]

CHEM_LINK_CHIR_ATOM

Data items in the CHEM_LINK_CHIR_ATOM category enumerate the atoms bonded to a chiral atom in a link between two chemical components.

Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_chir_atom.chir_id**
_chem_link_chir_atom.atom_id

_chem_link_chir_atom.atom_comp_id (*ucode*)
This data item indicates whether the atom bonded to a chiral atom is found in the first or the second of the two components connected by the link.

The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_chir_atom]

***_chem_link_chir_atom.atom_id** (*code*)
The ID of an atom bonded to the chiral atom. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

***_chem_link_chir_atom.chir_id**
This data item is a pointer to **_chem_link_chir.id** in the CHEM_LINK_CHIR category.

_chem_link_chir_atom.dev (*float*)
The standard uncertainty (estimated standard deviation) of the position of this atom from the plane defined by all of the atoms in the plane.

[chem_link_chir_atom]

CHEM_LINK_PLANE

Data items in the CHEM_LINK_PLANE category provide identifiers for the planes in a link between two chemical components. The atoms in the plane are specified in the CHEM_LINK_PLANE_ATOM category.

Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_plane.link_id**
_chem_link_plane.id

***_chem_link_plane.id** (*code*)
The value of **_chem_link_plane.id** must uniquely identify a record in the CHEM_LINK_PLANE list.

The following item(s) have an equivalent role in their respective categories:
_chem_link_plane_atom.plane_id. **[chem_link_plane]**

***_chem_link_plane.link_id**
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

_chem_link_plane.number_atoms_all (*int*)
The total number of atoms in the plane.

[chem_link_plane]

_chem_link_plane.number_atoms_nh (*int*)
The number of non-hydrogen atoms in the plane.

[chem_link_plane]

CHEM_LINK_PLANE_ATOM

Data items in the CHEM_LINK_PLANE_ATOM category enumerate the atoms in a plane in a link between two chemical components.

Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_plane_atom.plane_id**
_chem_link_plane_atom.atom_id

_chem_link_plane_atom.atom_comp_id (*ucode*)
This data item indicates whether the atom in a plane is found in the first or the second of the two components connected by the link.

The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_plane_atom]

***_chem_link_plane_atom.atom_id** (*code*)
The ID of an atom involved in the plane. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

***_chem_link_plane_atom.plane_id**
This data item is a pointer to **_chem_link_plane.id** in the CHEM_LINK_PLANE category.

CHEM_LINK_TOR

Data items in the CHEM_LINK_TOR category record details about the torsion angles in a link between two chemical components. As torsion angles can have more than one target value, the target values are specified in the CHEM_LINK_TOR_VALUE category.

Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_tor.link_id**
_chem_link_tor.id

_chem_link_tor.atom_1_comp_id (*ucode*)
This data item indicates whether atom 1 is found in the first or the second of the two components connected by the link.

The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_tor]

_chem_link_tor.atom_2_comp_id (*ucode*)
This data item indicates whether atom 2 is found in the first or the second of the two components connected by the link.

The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_tor]

_chem_link_tor.atom_3_comp_id (*ucode*)
This data item indicates whether atom 3 is found in the first or the second of the two components connected by the link.

The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_tor]

_chem_link_tor.atom_4_comp_id (*ucode*)
This data item indicates whether atom 4 is found in the first or the second of the two components connected by the link.
The data value must be one of the following:

1	the atom is in component 1
2	the atom is in component 2

[chem_link_tor]

*__chem_link_tor.atom_id_1__ (*code*)
The ID of the first of the four atoms that define the torsion angle. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

*__chem_link_tor.atom_id_2__ (*code*)
The ID of the second of the four atoms that define the torsion angle. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

*__chem_link_tor.atom_id_3__ (*code*)
The ID of the third of the four atoms that define the torsion angle. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

*__chem_link_tor.atom_id_4__ (*code*)
The ID of the fourth of the four atoms that define the torsion angle. As this data item does not point to a specific atom in a specific chemical component, it is not a child in the linkage sense.

*__chem_link_tor.id__ (*code*)
The value of **_chem_link_tor.id** must uniquely identify a record in the CHEM_LINK_TOR list.
The following item(s) have an equivalent role in their respective categories:
_chem_link_tor_value.tor_id. **[chem_link_tor]**

*__chem_link_tor.link_id__
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

CHEM_LINK_TOR_VALUE

Data items in the CHEM_LINK_TOR_VALUE category record details about the target values for the torsion angles enumerated in the CHEM_LINK_TOR list. Target values may be specified as angles in degrees, as a distance between the first and fourth atoms, or both.
Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_chem_link_tor_value.tor_id**

*__chem_link_tor_value.angle__ (*float, su*)
A value that should be taken as a potential target value for the torsion angle associated with the specified atoms, expressed in degrees.
The permitted range is $[-180.0, 180.0]$.
Related item: **_chem_link_tor_value.angle_esd** (associated esd).
[chem_link_tor_value]

*__chem_link_tor_value.angle_esd__ (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_link_tor_value.angle**.
The permitted range is $[-180.0, 180.0]$.
Related item: **_chem_link_tor_value.angle** (associated value).
[chem_link_tor_value]

_chem_link_tor_value.dist (*float, su*)
A value that should be taken as a potential target value for the torsion angle associated with the specified atoms, expressed as the distance between the atoms specified by **_chem_link_tor.atom_id_1** and **_chem_link_tor.atom_id_4** in the referenced record in the CHEM_LINK_TOR list. Note that the torsion angle cannot be fully specified by a distance (for instance, a torsion angle of -60° will yield the same distance as a 60° angle). However, the distance specification can be useful for refinement in situations in which the angle is already close to the desired value.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_link_tor_value.dist_esd** (associated esd).
[chem_link_tor_value]

_chem_link_tor_value.dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_chem_link_tor_value.dist**.
The permitted range is $[0.0, \infty)$.
Related item: **_chem_link_tor_value.dist** (associated value).
[chem_link_tor_value]

*__chem_link_tor_value.tor_id__
This data item is a pointer to **_chem_link_tor.id** in the CHEM_LINK_TOR category.

CHEMICAL

Data items in the CHEMICAL category would not in general be used in a macromolecular CIF. See instead the ENTITY data items. Data items in the CHEMICAL category record details about the composition and chemical properties of the compounds. The formula data items must agree with those that specify the density, unit-cell and Z values.
Category group(s): **inclusive_group**
chemical_group
Category key(s): **_chemical.entry_id**

*Example 1 – based on data set 9597gaus of Alyea, Ferguson & Kannan [Acta Cryst. (1996), **C52**, 765–767].*

```
_chemical.entry_id       '9597gaus'
_chemical.name_systematic
  trans-bis(tricyclohexylphosphine)tetracarbonylmolybdenum(0)
```

_chemical.absolute_configuration (*code*)
_chemical_absolute_configuration (*cif_core.dic 2.3*)
Necessary conditions for the assignment of **_chemical.absolute_configuration** are given by H. D. Flack and G. Bernardinelli (1999, 2000).

References: Flack, H. D. & Bernardinelli, G. (1999). *Acta Cryst.* A**55**, 908–915. Flack, H. D. & Bernardinelli, G. (2000). *J. Appl. Cryst.* **33**, 1143–1148.

The data value must be one of the following:

rm	Absolute configuration established by the structure determination of a compound containing a chiral reference molecule of known absolute configuration.
ad	Absolute configuration established by anomalous-dispersion effects in diffraction measurements on the crystal.
rmad	Absolute configuration established by the structure determination of a compound containing a chiral reference molecule of known absolute configuration and confirmed by anomalous-dispersion effects in diffraction measurements on the crystal.
syn	Absolute configuration has not been established by anomalous-dispersion effects in diffraction measurements on the crystal. The enantiomer has been assigned by reference to an unchanging chiral centre in the synthetic procedure.

unk Absolute configuration is unknown, there being no firm chemical evidence for its assignment to hand and it having not been established by anomalous-dispersion effects in diffraction measurements on the crystal. An arbitrary choice of enantiomer has been made.

. Inapplicable.

[chemical]

_chemical.compound_source (*text*)

_chemical_compound_source (*cif_core.dic 2.0.1*)

Description of the source of the compound under study, or of the parent molecule if a simple derivative is studied. This includes the place of discovery for minerals or the actual source of a natural product.

Examples: 'From Norilsk (USSR)',
'Extracted from the bark of Cinchona Naturalis'. **[chemical]**

*_chemical.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

_chemical.melting_point (*float*)

_chemical_melting_point (*cif_core.dic 2.0.1*)

The temperature in kelvins at which the crystalline solid changes to a liquid.

The permitted range is $[0.0, \infty)$. **[chemical]**

_chemical.melting_point_gt (*float*)

_chemical_melting_point_gt (*cif_core.dic 2.3*)

A temperature in kelvins above which the melting point (the temperature at which the crystalline solid changes to a liquid) lies. **_chemical.melting_point_gt** and **_chemical.melting_point_lt** allow a range of temperatures to be given. **_chemical.melting_point** should always be used in preference to these two items whenever possible.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.melting_point** (alternate). **[chemical]**

_chemical.melting_point_lt (*float*)

_chemical_melting_point_lt (*cif_core.dic 2.3*)

A temperature in kelvins below which the melting point (the temperature at which the crystalline solid changes to a liquid) lies. **_chemical.melting_point_gt** and **_chemical.melting_point_lt** allow a range of temperatures to be given. **_chemical.melting_point** should always be used in preference to these two items whenever possible.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.melting_point** (alternate). **[chemical]**

_chemical.name_common (*text*)

_chemical_name_common (*cif_core.dic 2.0.1*)

Trivial name by which the compound is commonly known.

Example: '1-bromoestradiol'. **[chemical]**

_chemical.name_mineral (*text*)

_chemical_name_mineral (*cif_core.dic 2.0.1*)

Mineral name accepted by the International Mineralogical Association. Use only for natural minerals. See also **_chemical.compound_source**.

Example: 'chalcopyrite'. **[chemical]**

_chemical.name_structure_type (*text*)

_chemical_name_structure_type (*cif_core.dic 2.0.1*)

Commonly used structure-type name. Usually only applied to minerals or inorganic compounds.

Examples: 'perovskite', 'sphalerite', 'A15'. **[chemical]**

_chemical.name_systematic (*text*)

_chemical_name_systematic (*cif_core.dic 2.0.1*)

IUPAC or *Chemical Abstracts* full name of the compound.

Example: '1-bromoestra-1,3,5(10)-triene-3,17\b-diol'. **[chemical]**

_chemical.optical_rotation (*line*)

_chemical_optical_rotation (*cif_core.dic 2.3*)

The optical rotation in solution of the compound is specified in the following format:

$$[\alpha]^{\mathrm{TEMP}}_{\mathrm{WAVE}} = \mathrm{SORT} \quad (c = \mathrm{CONC},\ \mathrm{SOLV}),$$

where TEMP is the temperature of the measurement in degrees Celsius, WAVE is an indication of the wavelength of the light used for the measurement, CONC is the concentration of the solution given as the mass of the substance in g per 100 ml of solution, SORT is the signed value (preceded by a + or a - sign) of $100\alpha/(lc)$, where α is the signed optical rotation in degrees measured in a cell of length l in dm and c is the value of CONC as defined above, and SOLV is the chemical formula of the solvent.

Example: '[\a]^25^~D~ = +108 (c = 3.42, CHCl~3~)'. **[chemical]**

_chemical.properties_biological (*text*)

_chemical_properties_biological (*cif_core.dic 2.3*)

A free-text description of the biological properties of the material.

Examples:

```
; diverse biological activities including use as a
  laxative and strong antibacterial activity against
  S. aureus and weak activity against
  cyclooxygenase-1 (COX-1)
;
; antibiotic activity against Bacillus subtilis
  (ATCC 6051) but no significant activity against
  Candida albicans (ATCC 14053), Aspergillus flavus
  (NRRL 6541) and Fusarium verticillioides (NRRL
  25457)
;
; weakly potent lipoxygenase nonredox inhibitor
;
; no influenza A virus sialidase inhibitory and
  plaque reduction activities
;
; low toxicity against Drosophila melanogaster
;
```

[chemical]

_chemical.properties_physical (*text*)

_chemical_properties_physical (*cif_core.dic 2.3*)

A free-text description of the physical properties of the material.

Examples: 'air-sensitive', 'moisture-sensitive', 'hygroscopic', 'deliquescent', 'oxygen-sensitive', 'photo-sensitive', 'pyrophoric', 'semiconductor', 'ferromagnetic at low temperature', 'paramagnetic and thermochromic'. **[chemical]**

_chemical.temperature_decomposition (*float, su*)

_chemical_temperature_decomposition (*cif_core.dic 2.3*)

The temperature in kelvins at which the solid decomposes.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.temperature_decomposition_esd** (associated esd) .

Example: '350'. **[chemical]**

_chemical.temperature_decomposition_esd (*float*)

The estimated standard deviation of **_chemical.temperature_decomposition**.

Related item: **_chemical.temperature_decomposition** (associated value).

[chemical]

_chemical.temperature_decomposition_gt (*float*)
_chemical_temperature_decomposition_gt (*cif_core.dic 2.3*)

A temperature in kelvins above which the solid is known to decompose. **_chemical.temperature_decomposition_gt** and **_chemical.temperature_decomposition_lt** allow a range of temperatures to be given. **_chemical.temperature_decomposition** should always be used in preference to these two items whenever possible.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.temperature_decomposition** (alternate) .

Example: '350'. **[chemical]**

_chemical.temperature_decomposition_lt (*float*)
_chemical_temperature_decomposition_lt (*cif_core.dic 2.3*)

A temperature in kelvins below which the solid is known to decompose. **_chemical.temperature_decomposition_gt** and **_chemical.temperature_decomposition_lt** allow a range of temperatures to be given. **_chemical.temperature_decomposition** should always be used in preference to these two items whenever possible.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.temperature_decomposition** (alternate) .

Example: '350'. **[chemical]**

_chemical.temperature_sublimation (*float, su*)
_chemical_temperature_sublimation (*cif_core.dic 2.3*)

The temperature in kelvins at which the solid sublimes.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.temperature_sublimation_esd** (associated esd) .

Example: '350'. **[chemical]**

_chemical.temperature_sublimation_esd (*float*)

The estimated standard deviation of **_chemical.temperature_sublimation**.

Related item: **_chemical.temperature_sublimation** (associated value).

[chemical]

_chemical.temperature_sublimation_gt (*float*)
_chemical_temperature_sublimation_gt (*cif_core.dic 2.3*)

A temperature in kelvins above which the solid is known to sublime. **_chemical.temperature_sublimation_gt** and **_chemical.temperature_sublimation_lt** allow a range of temperatures to be given. **_chemical.temperature_sublimation** should always be used in preference to these two items whenever possible.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.temperature_sublimation** (alternate) .

Example: '350'. **[chemical]**

_chemical.temperature_sublimation_lt (*float*)
_chemical_temperature_sublimation_lt (*cif_core.dic 2.3*)

A temperature in kelvins below which the solid is known to sublime. **_chemical.temperature_sublimation_gt** and **_chemical.temperature_sublimation_lt** allow a range of temperatures to be given. **_chemical.temperature_sublimation** should always be used in preference to these two items whenever possible.

The permitted range is $[0.0, \infty)$.

Related item: **_chemical.temperature_sublimation** (alternate) .

Example: '350'. **[chemical]**

CHEMICAL_CONN_ATOM

Data items in the CHEMICAL_CONN_ATOM category would not, in general, be used in a macromolecular CIF. See instead the ENTITY data items. Data items in the CHEMICAL_CONN_ATOM and CHEMICAL_CONN_BOND categories record details about the two-dimensional (2D) chemical structure of the molecular species. They allow a 2D chemical diagram to be reconstructed for use in a publication or in a database search for structural and substructural relationships. The CHEMICAL_CONN_ATOM data items provide information about the chemical properties of the atoms in the structure. In cases where crystallographic and molecular symmetry elements coincide, they must also contain symmetry-generated atoms, so that the CHEMICAL_CONN_ATOM and CHEMICAL_CONN_BOND data items will always describe a complete chemical entity.

Category group(s): **inclusive_group**
chemical_group

Category key(s): **_chemical_conn_atom.number**

*Example 1 – based on data set DPTD of Yamin, Suwandi, Fun, Sivakumar & bin Shawkataly [Acta Cryst. (1996), C**52**, 951–953].*

```
loop_
_chemical_conn_atom.number
_chemical_conn_atom.type_symbol
_chemical_conn_atom.display_x
_chemical_conn_atom.display_y
_chemical_conn_atom.NCA
_chemical_conn_atom.NH
    1   S    .39  .81   1   0
    2   S    .39  .96   2   0
    3   N    .14  .88   3   0
    4   C    .33  .88   3   0
    5   C    .11  .96   2   2
    6   C    .03  .96   2   2
    7   C    .03  .80   2   2
    8   C    .11  .80   2   2
    9   S    .54  .81   1   0
    10  S    .54  .96   2   0
    11  N    .80  .88   3   0
    12  C    .60  .88   3   0
    13  C    .84  .96   2   2
    14  C    .91  .96   2   2
    15  C    .91  .80   2   2
    16  C    .84  .80   2   2
```

_chemical_conn_atom.charge (*int*)
_chemical_conn_atom_charge (*cif_core.dic 2.0.1*)

The net integer charge assigned to this atom. This is the formal charge assignment normally found in chemical diagrams.

The permitted range is $[-8, 8]$. Where no value is given, the assumed value is '0'.

Examples: '1' (for an ammonium nitrogen), '-1' (for a chloride ion).

[chemical_conn_atom]

_chemical_conn_atom.display_x (*float*)
_chemical_conn_atom_display_x (*cif_core.dic 2.0.1*)

The 2D Cartesian *x* coordinate of the position of this atom in a recognizable chemical diagram. The coordinate origin is at the lower left corner, the *x* axis is horizontal and the *y* axis is vertical. The coordinates must lie in the range 0.0 to 1.0. These coordinates can be obtained from projections of a suitable uncluttered view of the molecular structure.

The permitted range is [0.0, 1.0]. **[chemical_conn_atom]**

_chemical_conn_atom.display_y (*float*)
_chemical_conn_atom_display_y (*cif_core.dic 2.0.1*)

The 2D Cartesian *y* coordinate of the position of this atom in a recognizable chemical diagram. The coordinate origin is at the lower left corner, the *x* axis is horizontal and the *y* axis is vertical. The coordinates must lie in the range 0.0 to 1.0. These coordinates can be obtained from projections of a suitable uncluttered view of the molecular structure.

The permitted range is [0.0, 1.0]. **[chemical_conn_atom]**

_chemical_conn_atom.NCA (*int*)
_chemical_conn_atom_NCA (*cif_core.dic 2.0.1*)
The number of connected atoms excluding terminal hydrogen atoms.
The permitted range is $[0, \infty)$. **[chemical_conn_atom]**

_chemical_conn_atom.NH (*int*)
_chemical_conn_atom_NH (*cif_core.dic 2.0.1*)
The total number of hydrogen atoms attached to this atom, regardless of whether they are included in the refinement or the ATOM_SITE list. This number is the same as **_atom_site.attached_hydrogens** only if none of the hydrogen atoms appear in the ATOM_SITE list.
The permitted range is $[0, \infty)$. **[chemical_conn_atom]**

***_chemical_conn_atom.number** (*int*)
_chemical_conn_atom_number (*cif_core.dic 2.0.1*)
The chemical sequence number to be associated with this atom. Within an ATOM_SITE list, this number must match one of the **_atom_site.chemical_conn_number** values.
The following item(s) have an equivalent role in their respective categories:
_atom_site.chemical_conn_number,
_chemical_conn_bond.atom_1,
_chemical_conn_bond.atom_2.
The permitted range is $[1, \infty)$. **[chemical_conn_atom]**

***_chemical_conn_atom.type_symbol**
_chemical_conn_atom_type_symbol (*cif_core.dic 2.0.1*)
This data item is a pointer to **_atom_type.symbol** in the ATOM_TYPE category.

CHEMICAL_CONN_BOND

Data items in the CHEMICAL_CONN_BOND category would not, in general, be used in a macromolecular CIF. See instead the ENTITY data items. Data items in the CHEMICAL_CONN_ATOM and CHEMICAL_CONN_BOND categories record details about the two-dimensional (2D) chemical structure of the molecular species. They allow a 2D chemical diagram to be reconstructed for use in a publication or in a database search for structural and substructural relationships. The CHEMICAL_CONN_BOND data items specify the connections between the atoms in the CHEMICAL_CONN_ATOM list and the nature of the chemical bond between these atoms.
Category group(s): **inclusive_group**
chemical_group
Category key(s): **_chemical_conn_bond.atom_1**
_chemical_conn_bond.atom_2

*Example 1 – based on data set DPTD of Yamin, Suwandi, Fun, Sivakumar & bin Shawkataly [Acta Cryst. (1996), C**52**, 951–953].*

```
loop_
_chemical_conn_bond.atom_1
_chemical_conn_bond.atom_2
_chemical_conn_bond.type
   4     1     doub      4     3     sing
   4     2     sing      5     3     sing
   6     5     sing      7     6     sing
   8     7     sing      8     3     sing
   10    2     sing      12    9     doub
   12    11    sing      12    10    sing
   13    11    sing      14    13    sing
   15    14    sing      16    15    sing
   16    11    sing      17    5     sing
   18    5     sing      19    6     sing
   20    6     sing      21    7     sing
   22    7     sing      23    8     sing
   24    8     sing      25    13    sing
   26    13    sing      27    14    sing
   28    14    sing      29    15    sing
   30    15    sing      31    16    sing
   32    16    sing
```

***_chemical_conn_bond.atom_1**
_chemical_conn_bond_atom_1 (*cif_core.dic 2.0.1*)
This data item is a pointer to **_chemical_conn_atom.number** in the CHEMICAL_CONN_ATOM category.

***_chemical_conn_bond.atom_2**
_chemical_conn_bond_atom_2 (*cif_core.dic 2.0.1*)
This data item is a pointer to **_chemical_conn_atom.number** in the CHEMICAL_CONN_ATOM category.

_chemical_conn_bond.type (*ucode*)
_chemical_conn_bond_type (*cif_core.dic 2.0.1*)
The chemical bond type associated with the connection between the two sites **_chemical_conn_bond.atom_1** and **_chemical_conn_bond.atom_2**.
The data value must be one of the following:

sing	single bond
doub	double bond
trip	triple bond
quad	quadruple bond
arom	aromatic bond
poly	polymeric bond
delo	delocalized double bond
pi	π bond

Where no value is given, the assumed value is 'sing'. **[chemical_conn_bond]**

CHEMICAL_FORMULA

Data items in the CHEMICAL_FORMULA category would not, in general, be used in a macromolecular CIF. See instead the ENTITY data items. Data items in the CHEMICAL_FORMULA category specify the composition and chemical properties of the compound. The formula data items must agree with those that specify the density, unit-cell and Z values. The following rules apply to the construction of the data items **_chemical_formula.analytical**, **_chemical_formula.structural** and **_chemical_formula.sum**. For the data item **_chemical_formula.moiety**, the formula construction is broken up into residues or moieties, *i.e.* groups of atoms that form a molecular unit or molecular ion. The rules given below apply within each moiety but different requirements apply to the way that moieties are connected (see **_chemical_formula.moiety**). (1) Only recognized element symbols may be used. (2) Each element symbol is followed by a 'count' number. A count of '1' may be omitted. (3) A space or parenthesis must separate each cluster of (element symbol + count). (4) Where a group of elements is enclosed in parentheses, the multiplier for the group must follow the closing parenthesis. That is, all element and group multipliers are assumed to be printed as subscripted numbers. (An exception to this rule exists for **_chemical_formula.moiety** formulae where pre- and post-multipliers are permitted for molecular units.) (5) Unless the elements are ordered in a manner that corresponds to their chemical structure, as in **_chemical_formula.structural**, the order of the elements within any group or moiety should be: C, then H, then the other elements in alphabetical order of their symbol. This is the 'Hill' system used by *Chemical Abstracts*. This ordering is used in **_chemical_formula.moiety** and **_chemical_formula.sum**.
Category group(s): **inclusive_group**
chemical_group
Category key(s): **_chemical_formula.entry_id**

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [(1991). Acta Cryst. C**47**, 2276–2277].*

```
_chemical_formula.entry_id          'TOZ'
_chemical_formula.moiety            'C18 H25 N O3'
_chemical_formula.sum               'C18 H25 N O3'
_chemical_formula.weight            303.40
```

_chemical_formula.analytical (*text*)

_chemical_formula_analytical (*cif_core.dic 2.0.1*)

Formula determined by standard chemical analysis including trace elements. See the CHEMICAL_FORMULA category description for rules for writing chemical formulae. Parentheses are used only for standard uncertainties (estimated standard deviations).

Example: 'Fe2.45(2) Ni1.60(3) S4'. **[chemical_formula]**

*_chemical_formula.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

_chemical_formula.iupac (*text*)

_chemical_formula_iupac (*cif_core.dic 2.0.1*)

Formula expressed in conformance with IUPAC rules for inorganic and metal-organic compounds where these conflict with the rules for any other CHEMICAL_FORMULA entries. Typically used for formatting a formula in accordance with journal rules. This should appear in the data block in addition to the most appropriate of the other CHEMICAL_FORMULA data names.

Reference: IUPAC (1990). *Nomenclature of Inorganic Chemistry*. Oxford: Blackwell Scientific Publications.

Example: '[Co Re (C12 H22 P)2 (C O)6].0.5C H3 O H'. **[chemical_formula]**

_chemical_formula.moiety (*text*)

_chemical_formula_moiety (*cif_core.dic 2.0.1*)

Formula with each discrete bonded residue or ion shown as a separate moiety. See the CHEMICAL_FORMULA category description for rules for writing chemical formulae. In addition to the general formulae requirements, the following rules apply: (1) Moieties are separated by commas ','. (2) The order of elements within a moiety follows general rule (5) in the CHEMICAL_FORMULA category description. (3) Parentheses are not used within moieties but may surround a moiety. Parentheses may not be nested. (4) Charges should be placed at the end of the moiety. The charge '+' or '-' may be preceded by a numerical multiplier and should be separated from the last (element symbol + count) by a space. Pre- or post-multipliers may be used for individual moieties.

Examples: 'C7 H4 Cl Hg N O3 S', 'C12 H17 N4 O S 1+, C6 H2 N3 O7 1-', 'C12 H16 N2 O6, 5(H2 O1)', '(Cd 2+)3, (C6 N6 Cr 3-)2, 2(H2 O)'. **[chemical_formula]**

_chemical_formula.structural (*text*)

_chemical_formula_structural (*cif_core.dic 2.0.1*)

See the CHEMICAL_FORMULA category description for the rules for writing chemical formulae for inorganics, organometallics, metal complexes *etc.*, in which bonded groups are preserved as discrete entities within parentheses, with post-multipliers as required. The order of the elements should give as much information as possible about the chemical structure. Parentheses may be used and nested as required. This formula should correspond to the structure as actually reported, *i.e.* trace elements not included in atom-type and atom-site data should not be included in this formula (see also **_chemical_formula.analytical**).

Examples: 'Ca ((Cl O3)2 O)2 (H2 O)6', '(Pt (N H3)2 (C5 H7 N3 O)2) (Cl O4)2'. **[chemical_formula]**

_chemical_formula.sum (*text*)

_chemical_formula_sum (*cif_core.dic 2.0.1*)

See the CHEMICAL_FORMULA category description for the rules for writing chemical formulae in which all discrete bonded residues and ions are summed over the constituent elements, following the ordering given in general rule (5) in the CHEMICAL_FORMULA category description. Parentheses are not normally used.

Example: 'C18 H19 N7 O8 S'. **[chemical_formula]**

_chemical_formula.weight (*float*)

_chemical_formula_weight (*cif_core.dic 2.0.1*)

Formula mass in daltons. This mass should correspond to the formulae given under **_chemical_formula.structural**, **_chemical_formula.moiety** or **_chemical_formula.sum** and, together with the Z value and cell parameters, should yield the density given as **_exptl_crystal.density_diffrn**.

The permitted range is $[1.0, \infty)$. **[chemical_formula]**

_chemical_formula.weight_meas (*float*)

_chemical_formula_weight_meas (*cif_core.dic 2.0.1*)

Formula mass in daltons measured by a non-diffraction experiment.

The permitted range is $[1.0, \infty)$. **[chemical_formula]**

CITATION

Data items in the CITATION category record details about the literature cited as being relevant to the contents of the data block.

Category group(s): **inclusive_group**
citation_group

Category key(s): **_citation.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_citation.id
_citation.coordinate_linkage
_citation.title
_citation.country
_citation.journal_abbrev
_citation.journal_volume
_citation.journal_issue
_citation.page_first
_citation.page_last
_citation.year
_citation.journal_id_ASTM
_citation.journal_id_ISSN
_citation.journal_id_CSD
_citation.book_title
_citation.book_publisher
_citation.book_id_ISBN
_citation.details
  primary  yes
; Crystallographic analysis of a complex between human
  immunodeficiency virus type 1 protease and acetyl-pepstatin
  at 2.0-Angstroms resolution.
;
  US  'J. Biol. Chem.'  265  .  14209  14219  1990
  HBCHA3  0021-9258  071  .  .  .
; The publication that directly relates to this coordinate
  set.
;
  2  no
; Three-dimensional structure of aspartyl-protease from human
  immunodeficiency virus HIV-1.
;
  UK  'Nature'  337  .  615  619  1989
  NATUAS  0028-0836  006  .  .  .
; Determination of the structure of the unliganded enzyme.
;
  3 no
; Crystallization of the aspartylprotease from human
  immunodeficiency virus, HIV-1.
;
  US  'J. Biol. Chem.'  264  .  1919  1921  1989
  HBCHA3  0021-9258  071  .  .  .
; Crystallization of the unliganded enzyme.
;
  4 no
; Human immunodeficiency virus protease. Bacterial expression
  and characterization of the purified aspartic protease.
;
  US  'J. Biol. Chem.'  264  .  2307  2312  1989
  HBCHA3  0021-9258  071  .  .  .
; Expression and purification of the enzyme.
;
```

_citation.abstract (*text*)
_citation_abstract (*cif_core.dic 2.0.1*)
Abstract for the citation. This is used most when the citation is extracted from a bibliographic database that contains full text or abstract information.
[citation]

_citation.abstract_id_CAS (*text*)
_citation_abstract_id_CAS (*cif_core.dic 2.0.1*)
The *Chemical Abstracts* Service (CAS) abstract identifier; relevant for journal articles.
[citation]

_citation.book_id_ISBN (*line*)
_citation_book_id_ISBN (*cif_core.dic 2.0.1*)
The International Standard Book Number (ISBN) code assigned to the book cited; relevant for books or book chapters.
[citation]

_citation.book_publisher (*text*)
_citation_book_publisher (*cif_core.dic 2.0.1*)
The name of the publisher of the citation; relevant for books or book chapters.
Example: 'John Wiley and Sons'. **[citation]**

_citation.book_publisher_city (*text*)
_citation_book_publisher_city (*cif_core.dic 2.0.1*)
The location of the publisher of the citation; relevant for books or book chapters.
Example: 'London'. **[citation]**

_citation.book_title (*text*)
_citation_book_title (*cif_core.dic 2.0.1*)
The title of the book in which the citation appeared; relevant for books or book chapters.
[citation]

_citation.coordinate_linkage (*ucode*)
_citation_coordinate_linkage (*cif_core.dic 2.0.1*)
_citation.coordinate_linkage states whether this citation is concerned with precisely the set of coordinates given in the data block. If, for instance, the publication described the same structure, but the coordinates had undergone further refinement prior to the creation of the data block, the value of this data item would be 'no'.
The data value must be one of the following:

no	citation unrelated to current coordinates
n	abbreviation for 'no'
yes	citation related to current coordinates
y	abbreviation for 'yes'

[citation]

_citation.country (*line*)
_citation_country (*cif_core.dic 2.0.1*)
The country of publication; relevant for books and book chapters.
[citation]

_citation.database_id_CSD (*code*)
_citation_database_id_CSD (*cif_core.dic 2.3*)
Identifier ('refcode') of the database record in the Cambridge Structural Database that contains details of the cited structure.
Example: 'LEKKUH'. **[citation]**

_citation.database_id_Medline (*int*)
_citation_database_id_Medline (*cif_core.dic 2.0.1*)
Accession number used by Medline to categorize a specific bibliographic entry.
The permitted range is $[1, \infty)$.
Example: '89064067'. **[citation]**

_citation.details (*text*)
_citation_special_details (*cif_core.dic 2.0.1*)
A description of special aspects of the relationship of the contents of the data block to the literature item cited.
Examples:

```
; citation relates to this precise
  coordinate set
;
; citation relates to earlier low-resolution
  structure
;
; citation relates to further refinement of
  structure reported in citation 2
;
```

[citation]

***_citation.id** (*code*)
_citation_id (*cif_core.dic 2.0.1*)
The value of **_citation.id** must uniquely identify a record in the CITATION list. The **_citation.id** 'primary' should be used to indicate the citation that the author(s) consider to be the most pertinent to the contents of the data block. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_citation_author.citation_id,
_citation_editor.citation_id,
_software.citation_id.
Examples: 'primary', '1', '2'. **[citation]**

_citation.journal_abbrev (*line*)
_citation_journal_abbrev (*cif_core.dic 2.0.1*)
Abbreviated name of the cited journal as given in the *Chemical Abstracts* Service *Source Index*.
Example: 'J. Mol. Biol.'. **[citation]**

_citation.journal_full (*text*)
_citation_journal_full (*cif_core.dic 2.0.1*)
Full name of the cited journal; relevant for journal articles.
Example: 'Journal of Molecular Biology'. **[citation]**

_citation.journal_id_ASTM (*line*)
_citation_journal_id_ASTM (*cif_core.dic 2.0.1*)
The American Society for Testing and Materials (ASTM) code assigned to the journal cited (also referred to as the CODEN designator of the *Chemical Abstracts* Service); relevant for journal articles.
[citation]

_citation.journal_id_CSD (*line*)
_citation_journal_id_CSD (*cif_core.dic 2.0.1*)
The Cambridge Structural Database (CSD) code assigned to the journal cited; relevant for journal articles. This is also the system used at the Protein Data Bank (PDB).
Example: '0070'. **[citation]**

_citation.journal_id_ISSN (*line*)
_citation_journal_id_ISSN (*cif_core.dic 2.0.1*)
The International Standard Serial Number (ISSN) code assigned to the journal cited; relevant for journal articles.
[citation]

_citation.journal_issue (*line*)
_citation_journal_issue (*cif_core.dic 2.0.1*)
Issue number of the journal cited; relevant for journal articles.
Example: '2'. **[citation]**

_citation.journal_volume (*line*)
_citation_journal_volume (*cif_core.dic 2.0.1*)
Volume number of the journal cited; relevant for journal articles.
Example: '174'. **[citation]**

_citation.language (*line*)
_citation_language (*cif_core.dic 2.0.1*)
Language in which the cited article is written.
Example: 'German'. **[citation]**

_citation.page_first (*line*)
_citation_page_first (*cif_core.dic 2.0.1*)
The first page of the citation; relevant for journal articles, books and book chapters.
[citation]

_citation.page_last (*line*)
_citation_page_last (*cif_core.dic 2.0.1*)
The last page of the citation; relevant for journal articles, books and book chapters.
[citation]

_citation.title (*text*)
_citation_title (*cif_core.dic 2.0.1*)
The title of the citation; relevant for journal articles, books and book chapters.
Example:
```
; Structure of diferric duck ovotransferrin at 2.35 \%A
resolution.
;
```
[citation]

_citation.year (*int*)
_citation_year (*cif_core.dic 2.0.1*)
The year of the citation; relevant for journal articles, books and book chapters.
Example: '1984'. **[citation]**

CITATION_AUTHOR

Data items in the CITATION_AUTHOR category record details about the authors associated with the citations in the CITATION list.
Category group(s): **inclusive_group**
citation_group
Category key(s): **_citation_author.citation_id**
_citation_author.name

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_citation_author.citation_id
_citation_author.ordinal
_citation_author.name
  primary  1  'Fitzgerald, P.M.D.'
  primary  2  'McKeever, B.M.'
  primary  3  'Van Middlesworth, J.F.'
  primary  4  'Springer, J.P.'
  primary  5  'Heimbach, J.C.'
  primary  6  'Leu, C.-T.'
  primary  7  'Herber, W.K.'
  primary  8  'Dixon, R.A.F.'
  primary  9  'Darke, P.L.'
  2        1  'Navia, M.A.'
  2        2  'Fitzgerald, P.M.D.'
  2        3  'McKeever, B.M.'
  2        4  'Leu, C.-T.'
  2        5  'Heimbach, J.C.'
  2        6  'Herber, W.K.'
  2        7  'Sigal, I.S.'
  2        8  'Darke, P.L.'
  2        9  'Springer, J.P.'
  3        1  'McKeever, B.M.'
  3        2  'Navia, M.A.'
  3        3  'Fitzgerald, P.M.D.'
  3        4  'Springer, J.P.'
  3        5  'Leu, C.-T.'
  3        6  'Heimbach, J.C.'
  3        7  'Herber, W.K.'
  3        8  'Sigal, I.S.'
  3        9  'Darke, P.L.'
  4        1  'Darke, P.L.'
  4        2  'Leu, C.-T.'
  4        3  'Davis, L.J.'
  4        4  'Heimbach, J.C.'
  4        5  'Diehl, R.E.'
  4        6  'Hill, W.S.'
  4        7  'Dixon, R.A.F.'
  4        8  'Sigal, I.S.'
```

***_citation_author.citation_id**
_citation_author_citation_id (*cif_core.dic 2.0.1*)
This data item is a pointer to **_citation.id** in the CITATION category.

***_citation_author.name** (*line*)
_citation_author_name (*cif_core.dic 2.0.1*)
Name of an author of the citation; relevant for journal articles, books and book chapters. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).
Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.'. **[citation_author]**

_citation_author.ordinal (*int*)
_citation_author_ordinal (*cif_core.dic 2.0.1*)
This data item defines the order of the author's name in the list of authors of a citation.
[citation_author]

CITATION_EDITOR

Data items in the CITATION_EDITOR category record details about the editors associated with the books or book chapters cited in the CITATION list.

Category group(s): **inclusive_group**
citation_group
Category key(s): **_citation_editor.citation_id**
_citation_editor.name

Example 1 – hypothetical example.

```
loop_
_citation_editor.citation_id
_citation_editor.name
  5        'McKeever, B.M.'
  5        'Navia, M.A.'
  5        'Fitzgerald, P.M.D.'
  5        'Springer, J.P.'
```

*_citation_editor.citation_id

_citation_editor_citation_id (*cif_core.dic 2.0.1*)

This data item is a pointer to **_citation.id** in the CITATION category.

_citation_editor.name (*line*)

_citation_editor_name (*cif_core.dic 2.0.1*)

Names of an editor of the citation; relevant for books and book chapters. The family name(s), followed by a comma and including any dynastic components, precedes the first name(s) or initial(s).

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.'. **[citation_editor]**

_citation_editor.ordinal (*int*)

_citation_editor_ordinal (*cif_core.dic 2.0.1*)

This data item defines the order of the editor's name in the list of editors of a citation.

[citation_editor]

COMPUTING

Data items in the COMPUTING category record details about the computer programs used in the crystal structure analysis. Data items in this category would not, in general, be used in a macromolecular CIF. The category SOFTWARE, which allows a more detailed description of computer programs and their attributes to be given, would be used instead.

Category group(s): **inclusive_group**
computing_group
Category key(s): **_computing.entry_id**

Example 1 – Rodrìguez-Romera, Ruiz-Pérez & Solans [Acta Cryst. (1996), C52, 1415–1417].

```
_computing.data_collection      'CAD-4 (Enraf-Nonius, 1989)'
_computing.cell_refinement      'CAD-4 (Enraf-Nonius, 1989)'
_computing.data_reduction       'CFEO (Solans, 1978)'
_computing.structure_solution   'SHELXS86 (Sheldrick, 1990)'
_computing.structure_refinement 'SHELXL93 (Sheldrick, 1993)'
_computing.molecular_graphics   'ORTEPII (Johnson, 1976)'
_computing.publication_material 'PARST (Nardelli, 1983)'
```

_computing.cell_refinement (*text*)

_computing_cell_refinement (*cif_core.dic 2.0.1*)

Software used for cell refinement. Give the program or package name and a brief reference.

Example: 'CAD4 (Enraf-Nonius, 1989)'. **[computing]**

_computing.data_collection (*text*)

_computing_data_collection (*cif_core.dic 2.0.1*)

Software used for data collection. Give the program or package name and a brief reference.

Example: 'CAD4 (Enraf-Nonius, 1989)'. **[computing]**

_computing.data_reduction (*text*)

_computing_data_reduction (*cif_core.dic 2.0.1*)

Software used for data reduction. Give the program or package name and a brief reference.

Example: 'DIFDAT, SORTRF, ADDREF (Hall & Stewart, 1990)'.
[computing]

*_computing.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

_computing.molecular_graphics (*text*)

_computing_molecular_graphics (*cif_core.dic 2.0.1*)

Software used for molecular graphics. Give the program or package name and a brief reference.

Example: 'FRODO (Jones, 1986), ORTEP (Johnson, 1965)'. **[computing]**

_computing.publication_material (*text*)

_computing_publication_material (*cif_core.dic 2.0.1*)

Software used for generating material for publication. Give the program or package name and a brief reference.

[computing]

_computing.structure_refinement (*text*)

_computing_structure_refinement (*cif_core.dic 2.0.1*)

Software used for refinement of the structure. Give the program or package name and a brief reference.

Examples: 'SHELX85 (Sheldrick, 1985)', 'X-PLOR (Brunger, 1992)'.
[computing]

_computing.structure_solution (*text*)

_computing_structure_solution (*cif_core.dic 2.0.1*)

Software used for solution of the structure. Give the program or package name and a brief reference.

Example: 'SHELX85 (Sheldrick, 1985)'. **[computing]**

DATABASE

Data items in the DATABASE category have been superseded by data items in the DATABASE_2 category. They are included here only for compliance with older CIFs.

Category group(s): **inclusive_group**
compliance_group
Category key(s): **_database.entry_id**

_database.code_CAS (*line*)

_database_code_CAS (*cif_core.dic 2.3*)

The code assigned by *Chemical Abstracts*.

[database]

_database.code_CSD (*line*)

_database_code_CSD (*cif_core.dic 2.3*)

The code assigned by the Cambridge Structural Database.

[database]

_database.code_ICSD (*line*)

_database_code_ICSD (*cif_core.dic 2.3*)

The code assigned by the Inorganic Crystal Structure Database.

[database]

_database.code_MDF (*line*)
_database_code_MDF (*cif_core.dic 2.3*)
The code assigned by the Metals Data File.
[database]

_database.code_NBS (*line*)
_database_code_NBS (*cif_core.dic 2.3*)
The code assigned by the NBS (NIST) Crystal Data Database.
[database]

_database.code_PDB (*line*)
_database_code_PDB (*cif_core.dic 2.3*)
The code assigned by the Protein Data Bank.
[database]

_database.code_PDF (*line*)
_database_code_PDF (*cif_core.dic 2.3*)
The code assigned by the Powder Diffraction File (JCPDS/ICDD).
[database]

_database.code_depnum_ccdc_archive (*line*)
_database_code_depnum_ccdc_archive (*cif_core.dic 2.3*)
Deposition numbers assigned by the Cambridge Crystallographic Data Centre (CCDC) to files containing structural information archived by the CCDC.
[database]

_database.code_depnum_ccdc_fiz (*line*)
_database_code_depnum_ccdc_fiz (*cif_core.dic 2.3*)
Deposition numbers assigned by the Fachinformationszentrum Karlsruhe (FIZ) to files containing structural information archived by the Cambridge Crystallographic Data Centre (CCDC).
[database]

_database.code_depnum_ccdc_journal (*line*)
_database_code_depnum_ccdc_journal (*cif_core.dic 2.3*)
Deposition numbers assigned by various journals to files containing structural information archived by the Cambridge Crystallographic Data Centre (CCDC).
[database]

_database.CSD_history (*text*)
_database_CSD_history (*cif_core.dic 2.3*)
A history of changes made by the Cambridge Crystallographic Data Centre and incorporated into the Cambridge Structural Database (CSD).
[database]

***_database.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_database.journal_ASTM (*line*)
_database_journal_ASTM (*cif_core.dic 2.0.1*)
The ASTM CODEN designator for a journal as given in the *Chemical Source List* maintained by the *Chemical Abstracts* Service.
[database]

_database.journal_CSD (*line*)
_database_journal_CSD (*cif_core.dic 2.0.1*)
The journal code used in the Cambridge Structural Database.
[database]

DATABASE_2

Data items in the DATABASE_2 category record details about the database identifiers of the data block. These data items are assigned by database managers and should only appear in a data block if they originate from that source. The name of this category, DATABASE_2, arose because the category name DATABASE was already in use in the core CIF dictionary, but was used differently from the way it needed to be used in the mmCIF dictionary. Since CIF data names cannot be changed once they have been adopted, a new category had to be created.

Category group(s): **inclusive_group**
database_group
Category key(s): **_database_2.database_id**
_database_2.database_code

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_database_2.database_id       'PDB'
_database_2.database_code     '5HVP'
```

***_database_2.database_code** (*line*)
The code assigned by the database identified in **_database_2.database_id**.
Related items: **_database.code_CAS** (replaces),
_database.code_CSD (replaces),
_database.code_ICSD (replaces),
_database.code_MDF (replaces),
_database.code_NBS (replaces),
_database.code_PDF (replaces) .
Examples: '1ABC', 'ABCDEF'. **[database_2]**

***_database_2.database_id** (*ucode*)
An abbreviation that identifies the database.
Related items: **_database.code_CAS** (replaces),
_database.code_CSD (replaces),
_database.code_ICSD (replaces),
_database.code_MDF (replaces),
_database.code_NBS (replaces),
_database.code_PDF (replaces).
The data value must be one of the following:

CAS	*Chemical Abstracts*
CSD	Cambridge Structural Database (organic and metal-organic compounds)
ICSD	Inorganic Crystal Structure Database
MDF	Metals Data File (metal structures)
NDB	Nucleic Acid Database
NBS	NBS (NIST) Crystal Data Database (lattice parameters)
PDB	Protein Data Bank
PDF	Powder Diffraction File (JCPDS/ICDD)
RCSB	Research Collaboratory for Structural Bioinformatics
EBI	European Bioinformatics Institute

[database_2]

DATABASE_PDB_CAVEAT

Data items in the DATABASE_PDB_CAVEAT category record details about features of the data block flagged as 'caveats' by the Protein Data Bank (PDB). These data items are included only for consistency with PDB format files. They should appear in a data block only if that data block was created by reformatting a PDB format file.

Category group(s): **inclusive_group**
database_group
pdb_group
Category key(s): **_database_PDB_caveat.id**

Example 1 – hypothetical example.

```
loop_
_database_PDB_caveat.id
_database_PDB_caveat.text
1
; THE CRYSTAL TRANSFORMATION IS IN ERROR BUT IS
;
2
; UNCORRECTABLE AT THIS TIME
;
```

* **_database_PDB_caveat.id** (*int*)
A unique identifier for the PDB caveat record.

[database_PDB_caveat]

_database_PDB_caveat.text (*text*)
The full text of the PDB caveat record.

[database_PDB_caveat]

DATABASE_PDB_MATRIX

The DATABASE_PDB_MATRIX category provides placeholders for transformation matrices and vectors used by the Protein Data Bank (PDB). These data items are included only for consistency with older PDB format files. They should appear in a data block only if that data block was created by reformatting a PDB format file.

Category group(s): **inclusive_group**
database_group
pdb_group
Category key(s): **_database_PDB_matrix.entry_id**

* **_database_PDB_matrix.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_database_PDB_matrix.origx[1][1] (*float*)
The [1][1] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '1.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[1][2] (*float*)
The [1][2] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[1][3] (*float*)
The [1][3] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[2][1] (*float*)
The [2][1] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[2][2] (*float*)
The [2][2] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '1.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[2][3] (*float*)
The [2][3] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[3][1] (*float*)
The [3][1] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[3][2] (*float*)
The [3][2] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx[3][3] (*float*)
The [3][3] element of the PDB ORIGX matrix.
Where no value is given, the assumed value is '1.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx_vector[1] (*float*)
The [1] element of the PDB ORIGX vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx_vector[2] (*float*)
The [2] element of the PDB ORIGX vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.origx_vector[3] (*float*)
The [3] element of the PDB ORIGX vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[1][1] (*float*)
The [1][1] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '1.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[1][2] (*float*)
The [1][2] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[1][3] (*float*)
The [1][3] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[2][1] (*float*)
The [2][1] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[2][2] (*float*)
The [2][2] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '1.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[2][3] (*float*)
The [2][3] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[3][1] (*float*)
The [3][1] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[3][2] (*float*)
The [3][2] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale[3][3] (*float*)
The [3][3] element of the PDB SCALE matrix.
Where no value is given, the assumed value is '1.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale_vector[1] (*float*)
The [1] element of the PDB SCALE vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale_vector[2] (*float*)
The [2] element of the PDB SCALE vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

_database_PDB_matrix.scale_vector[3] (*float*)
The [3] element of the PDB SCALE vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_matrix]**

DATABASE_PDB_REMARK

Data items in the DATABASE_PDB_REMARK category record details about the data block as archived by the Protein Data Bank (PDB). Some data appearing in PDB REMARK records can be algorithmically extracted into the appropriate data items in the data block. These data items are included only for consistency with older PDB format files. They should appear in a data block only if that data block was created by reformatting a PDB format file.
Category group(s): **inclusive_group**
database_group
pdb_group
Category key(s): **_database_PDB_remark.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_database_PDB_remark.id
_database_PDB_remark.text
3
; REFINEMENT. BY THE RESTRAINED LEAST-SQUARES PROCEDURE OF J.
  KONNERT AND W. HENDRICKSON (PROGRAM *PROLSQ*).  THE R
  VALUE IS 0.176 FOR 12901 REFLECTIONS IN THE RESOLUTION
  RANGE 8.0 TO 2.0 ANGSTROMS WITH I .GT. SIGMA(I).

  RMS DEVIATIONS FROM IDEAL VALUES (THE VALUES OF
  SIGMA, IN PARENTHESES, ARE THE INPUT ESTIMATED
  STANDARD DEVIATIONS THAT DETERMINE THE RELATIVE
  WEIGHTS OF THE CORRESPONDING RESTRAINTS)
  DISTANCE RESTRAINTS (ANGSTROMS)
  BOND DISTANCE                              0.018(0.020)
  ANGLE DISTANCE                             0.038(0.030)
  PLANAR 1-4 DISTANCE                        0.043(0.040)
  PLANE RESTRAINT (ANGSTROMS)                  0.015(0.020)
  CHIRAL-CENTER RESTRAINT (ANGSTROMS**3)     0.177(0.150)
  NON-BONDED CONTACT RESTRAINTS (ANGSTROMS)
  SINGLE TORSION CONTACT                     0.216(0.500)
  MULTIPLE TORSION CONTACT                   0.207(0.500)
  POSSIBLE HYDROGEN BOND                     0.245(0.500)
  CONFORMATIONAL TORSION ANGLE RESTRAINT (DEGREES)
  PLANAR (OMEGA)                               2.6(3.0)
  STAGGERED                                   17.4(15.0)
  ORTHONORMAL                                 18.1(20.0)
;
4
; THE TWO CHAINS OF THE DIMERIC ENZYME HAS BEEN ASSIGNED THE
  THE CHAIN INDICATORS *A* AND *B*.
;
#  - - - - data truncated for brevity - - - -
```

***_database_PDB_remark.id** (*int*)
A unique identifier for the PDB remark record.
[database_PDB_remark]

_database_PDB_remark.text (*text*)
The full text of the PDB remark record.
[database_PDB_remark]

DATABASE_PDB_REV

Data items in the DATABASE_PDB_REV category record details about the history of the data block as archived by the Protein Data Bank (PDB). These data items are assigned by the PDB database managers and should only appear in a data block if they originate from that source.
Category group(s): **inclusive_group**
database_group
pdb_group
Category key(s): **_database_PDB_rev.num**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_database_PDB_rev.num
_database_PDB_rev.author_name
_database_PDB_rev.date
_database_PDB_rev.date_original
_database_PDB_rev.status
_database_PDB_rev.mod_type
  1  'Fitzgerald, Paula M.D' 1991-10-15  1990-04-30
  'full release'  0
```

_database_PDB_rev.author_name (*line*)
The name of the person responsible for submitting this revision to the PDB. The family name(s) followed by a comma precedes the first name(s) or initial(s).
Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.'. **[database_PDB_rev]**

_database_PDB_rev.date (*yyyy-mm-dd*)
Date the PDB revision took place. Taken from the REVDAT record.
[database_PDB_rev]

_database_PDB_rev.date_original (*yyyy-mm-dd*)
Date the entry first entered the PDB database in the form *yyyy-mm-dd*. Taken from the PDB HEADER record.
Example: '1980-08-21'. **[database_PDB_rev]**

_database_PDB_rev.mod_type (*int*)
Taken from the REVDAT record. Refer to the Protein Data Bank format description at http://www.rcsb.org/pdb/docs/format/pdbguide2.2/guide2.2_frame.html for details.
The data value must be one of the following:

0	initial entry
1	all other types of modification
2	modifications to CONECT records
3	modifications affecting the coordinates or their transforms (CRYST1, ORIGX, SCALE, MTRIX, TVECT, ATOM, HETATM, SIGATM records)
4	layer 1 to layer 2 revision which may affect all record types
5	data uniformity processing

[database_PDB_rev]

***_database_PDB_rev.num** (*int*)
The value of **_database_PDB_rev.num** must uniquely and sequentially identify a record in the DATABASE_PDB_REV list. Note that this item must be a number and that modification numbers are assigned in increasing numerical order.
The following item(s) have an equivalent role in their respective categories:
_database_PDB_rev_record.rev_num. **[database_PDB_rev]**

_database_PDB_rev.replaced_by (*line*)
The PDB code for a subsequent PDB entry that replaced the PDB file corresponding to this data block.
[database_PDB_rev]

_database_PDB_rev.replaces (*line*)
The PDB code for a previous PDB entry that was replaced by the PDB file corresponding to this data block.
[database_PDB_rev]

_database_PDB_rev.status (*uline*)
The status of this revision.
The data value must be one of the following:
'in preparation'
prerelease
'full release'
obsolete
[database_PDB_rev]

DATABASE_PDB_REV_RECORD

Data items in the DATABASE_PDB_REV_RECORD category record details about specific record types that were changed in a given revision of a PDB entry. These data items are assigned by the PDB database managers and should only appear in a data block if they originate from that source.
Category group(s): **inclusive_group**
database_group
pdb_group
Category key(s): **_database_PDB_rev_record.rev_num**
_database_PDB_rev_record.type

Example 1 – hypothetical example.

```
loop_
_database_PDB_rev_record.rev_num
_database_PDB_rev_record.type
_database_PDB_rev_record.details
  1  CONECT
; Error fix - incorrect connection between
  atoms 2312 and 2317
;
  2  MATRIX  'For consistency with 1995-08-04 style-guide'
  3  ORIGX   'Based on new data from author'
```

_database_PDB_rev_record.details (*text*)
A description of special aspects of the revision of records in this PDB entry.
Examples: 'Based on new data from author',
'For consistency with 1995-08-04 style-guide',
'For consistency with structural class'.
[database_PDB_rev_record]

***_database_PDB_rev_record.rev_num**
This data item is a pointer to **_database_PDB_rev.num** in the DATABASE_PDB_REV category.

***_database_PDB_rev_record.type** (*line*)
The types of records that were changed in this revision to a PDB entry.
Examples: 'CRYST1', 'SCALE', 'MTRIX', 'ATOM', 'HETATM'.
[database_PDB_rev_record]

DATABASE_PDB_TVECT

The DATABASE_PDB_TVECT category provides placeholders for the TVECT matrices and vectors used by the Protein Data Bank (PDB). These data items are included only for consistency with older PDB format files. They should appear in a data block only if the data block was created by reformatting a PDB format file.
Category group(s): **inclusive_group**
database_group
pdb_group
Category key(s): **_database_PDB_tvect.id**

_database_PDB_tvect.details (*text*)
A description of special aspects of this TVECT.
[database_PDB_tvect]

***_database_PDB_tvect.id** (*code*)
The value of **_database_PDB_tvect.id** must uniquely identify a record in the DATABASE_PDB_TVECT list. Note that this item need not be a number; it can be any unique identifier.
[database_PDB_tvect]

_database_PDB_tvect.vector[1] (*float*)
The [1] element of the PDB TVECT vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_tvect]**

_database_PDB_tvect.vector[2] (*float*)
The [2] element of the PDB TVECT vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_tvect]**

_database_PDB_tvect.vector[3] (*float*)
The [3] element of the PDB TVECT vector.
Where no value is given, the assumed value is '0.0'. **[database_PDB_tvect]**

DIFFRN

Data items in the DIFFRN category record details about the diffraction data and their measurement.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn.id                          'Set1'
_diffrn.ambient_temp                 293(3)
_diffrn.ambient_environment
; Mother liquor from the reservoir of the vapor diffusion
  experiment, mounted in room air
;
_diffrn.crystal_support
; 0.7 mm glass capillary, sealed with dental wax
;
_diffrn.crystal_treatment
; Equilibrated in rotating anode radiation enclosure for
  18 hours prior to beginning of data collection
;
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [(1991). Acta Cryst. C**47**, 2276–2277].*

```
_diffrn.id              'd1'
_diffrn.details
; \q scan width (1.0 + 0.14tan\q)\%, \q scan rate 1.2\% per
   min. Background counts for 5 sec on each side every scan.
;
_diffrn.ambient_temp     293
```

_diffrn.ambient_environment (*line*)
_diffrn_ambient_environment (*cif_core.dic 2.0.1*)
The gas or liquid surrounding the sample, if not air.
[diffrn]

_diffrn.ambient_pressure (*float, su*)
_diffrn_ambient_pressure (*cif_core.dic 2.3*)
The mean hydrostatic pressure in kilopascals at which the intensities were measured.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn.ambient_pressure_esd** (associated esd). **[diffrn]**

_diffrn.ambient_pressure_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_diffrn.ambient_pressure**.
Related item: **_diffrn.ambient_pressure** (associated value). **[diffrn]**

_diffrn.ambient_pressure_gt (*float*)
_diffrn_ambient_pressure_gt (*cif_core.dic 2.3*)
The mean hydrostatic pressure in kilopascals above which the intensities were measured. **_diffrn.ambient_pressure_gt** and **_diffrn.ambient_pressure_lt** allow a pressure range to be given. **_diffrn.ambient_pressure** should always be used in preference to these two items whenever possible.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn.ambient_pressure** (alternate). **[diffrn]**

_diffrn.ambient_pressure_lt (*float*)
_diffrn_ambient_pressure_lt (*cif_core.dic 2.3*)
The mean hydrostatic pressure in kilopascals below which the intensities were measured. **_diffrn.ambient_pressure_gt** and **_diffrn.ambient_pressure_lt** allow a pressure range to be given. **_diffrn.ambient_pressure** should always be used in preference to these two items whenever possible.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn.ambient_pressure** (alternate). **[diffrn]**

_diffrn.ambient_temp (*float, su*)
_diffrn_ambient_temperature (*cif_core.dic 2.0.1*)
The mean temperature in kelvins at which the intensities were measured.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn.ambient_temp_esd** (associated esd). **[diffrn]**

_diffrn.ambient_temp_details (*text*)
A description of special aspects of temperature control during data collection.
[diffrn]

_diffrn.ambient_temp_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_diffrn.ambient_temp**.
Related item: **_diffrn.ambient_temp** (associated value). **[diffrn]**

_diffrn.ambient_temp_gt (*float*)
_diffrn_ambient_temperature_gt (*cif_core.dic 2.3*)
The mean temperature in kelvins above which the intensities were measured. **_diffrn.ambient_temp_gt** and **_diffrn.ambient_temp_lt** allow a range of temperatures to be given. **_diffrn.ambient_temp** should always be used in preference to these two items whenever possible.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn.ambient_temp** (alternate). **[diffrn]**

_diffrn.ambient_temp_lt (*float*)
_diffrn_ambient_temperature_lt (*cif_core.dic 2.3*)
The mean temperature in kelvins below which the intensities were measured. **_diffrn.ambient_temp_gt** and **_diffrn.ambient_temp_lt** allow a range of temperatures to be given. **_diffrn.ambient_temp** should always be used in preference to these two items whenever possible.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn.ambient_temp** (alternate). **[diffrn]**

***_diffrn.crystal_id**
_diffrn_refln_crystal_id (*cif_core.dic 2.0.1*)
This data item is a pointer to **_exptl_crystal.id** in the EXPTL_CRYSTAL category.

_diffrn.crystal_support (*text*)
The physical device used to support the crystal during data collection.
Examples: 'glass capillary', 'quartz capillary', 'fiber', 'metal loop'.
[diffrn]

_diffrn.crystal_treatment (*text*)
_diffrn_crystal_treatment (*cif_core.dic 2.0.1*)
Remarks about how the crystal was treated prior to intensity measurement. Particularly relevant when intensities were measured at low temperature.
Examples: 'equilibrated in hutch for 24 hours',
'flash frozen in liquid nitrogen',
'slow cooled with direct air stream'. **[diffrn]**

_diffrn.details (*text*)
_diffrn_special_details (*cif_core.dic 2.0.1*)
Special details of the diffraction measurement process. Should include information about source instability, crystal motion, degradation and so on.
[diffrn]

***_diffrn.id** (*code*)
This data item uniquely identifies a set of diffraction data.
The following item(s) have an equivalent role in their respective categories:
_diffrn_detector.diffrn_id,
_diffrn_measurement.diffrn_id,
_diffrn_orient_matrix.diffrn_id,
_diffrn_orient_refln.diffrn_id,
_diffrn_radiation.diffrn_id,
_diffrn_refln.diffrn_id,
_diffrn_reflns.diffrn_id,
_diffrn_source.diffrn_id,
_diffrn_standard_refln.diffrn_id,
_diffrn_standards.diffrn_id. **[diffrn]**

DIFFRN_ATTENUATOR

Data items in the DIFFRN_ATTENUATOR category record details about the diffraction attenuator scales employed.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_attenuator.code**

Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276-2277].

```
_diffrn_attenuator.code       1
_diffrn_attenuator.scale      16.976
```

***_diffrn_attenuator.code** (*code*)
_diffrn_attenuator_code (*cif_core.dic 2.0.1*)
A code associated with a particular attenuator setting. This code is referenced by the **_diffrn_refln.attenuator_code** which is stored with the diffraction data. See **_diffrn_attenuator.scale**.
[diffrn_attenuator]

_diffrn_attenuator.material (*text*)
_diffrn_attenuator_material (*cif_core.dic 2.3*)
Material from which the attenuator is made.
[diffrn_attenuator]

_diffrn_attenuator.scale (*float*)

_diffrn_attenuator_scale (*cif_core.dic 2.0.1*)

The scale factor applied when an intensity measurement is reduced by an attenuator identified by **_diffrn_attenuator.code**. The measured intensity must be multiplied by this scale to convert it to the same scale as unattenuated intensities.

The permitted range is $[1.0, \infty)$. **[diffrn_attenuator]**

DIFFRN_DETECTOR

Data items in the DIFFRN_DETECTOR category describe the detector used to measure the scattered radiation, including any analyser and post-sample collimation.

Category group(s): **inclusive_group**
diffrn_group

Category key(s): **_diffrn_detector.diffrn_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_detector.diffrn_id            'd1'
_diffrn_detector.detector             'multiwire'
_diffrn_detector.type                 'Siemens'
```

_diffrn_detector.area_resol_mean (*float*)

_diffrn_detector_area_resol_mean (*cif_core.dic 2.3*)

The resolution of an area detector, in pixels mm^{-1}.

The permitted range is $[0.0, \infty)$. **[diffrn_detector]**

_diffrn_detector.details (*text*)

_diffrn_detector_details (*cif_core.dic 2.0.1*)

A description of special aspects of the radiation detector.

[diffrn_detector]

_diffrn_detector.detector (*text*)

_diffrn_radiation_detector (*cifdic.c91 1.0*)
_diffrn_detector (*cif_core.dic 2.0*)

The general class of the radiation detector.

Examples: 'photographic film', 'scintillation counter', 'CCD plate', 'BF~3~ counter'. **[diffrn_detector]**

*_diffrn_detector.diffrn_id

This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_detector.dtime (*float*)

_diffrn_detector_dtime (*cif_core.dic 2.3*)

The deadtime in microseconds of the detector used to measure the diffraction intensities.

The permitted range is $[0.0, \infty)$. **[diffrn_detector]**

_diffrn_detector.type (*text*)

_diffrn_detector_type (*cif_core.dic 2.0.1*)

The make, model or name of the detector device used.

[diffrn_detector]

DIFFRN_MEASUREMENT

Data items in the DIFFRN_MEASUREMENT category record details about the device used to orient and/or position the crystal during data measurement and the manner in which the diffraction data were measured.

Category group(s): **inclusive_group**
diffrn_group

Category key(s): **_diffrn_measurement.diffrn_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_measurement.diffrn_id         'd1'
_diffrn_measurement.device            '3-circle camera'
_diffrn_measurement.device_type       'Supper model x'
_diffrn_measurement.device_details    'none'
_diffrn_measurement.method            'omega scan'
_diffrn_measurement.details
; 440 frames, 0.20 degrees, 150 sec, detector distance 12 cm,
  detector angle 22.5 degrees
;
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
_diffrn_measurement.diffrn_id      's1'
_diffrn_measurement.device_type
     'Philips PW1100/20 diffractometer'
_diffrn_measurement.method         \q/2\q
```

_diffrn_measurement.details (*text*)

_diffrn_measurement_details (*cif_core.dic 2.0.1*)

A description of special aspects of the intensity measurement.

Example:

```
; 440 frames, 0.20 degrees, 150 sec, detector
  distance 12 cm, detector angle 22.5 degrees
;
```

[diffrn_measurement]

_diffrn_measurement.device (*text*)

_diffrn_measurement_device (*cif_core.dic 2.0.1*)

The general class of goniometer or device used to support and orient the specimen.

Examples: '3-circle camera', '4-circle camera', 'kappa-geometry camera', 'oscillation camera', 'precession camera'. **[diffrn_measurement]**

_diffrn_measurement.device_details (*text*)

_diffrn_measurement_device_details (*cif_core.dic 2.0.1*)

A description of special aspects of the device used to measure the diffraction intensities.

Example:

```
; commercial goniometer modified locally to
  allow for 90\% \t arc
;
```

[diffrn_measurement]

_diffrn_measurement.device_type (*text*)

_diffrn_measurement_device_type (*cif_core.dic 2.0.1*)

The make, model or name of the measurement device (goniometer) used.

Examples: 'Supper model q', 'Huber model r', 'Enraf-Nonius model s', 'homemade'. **[diffrn_measurement]**

*_diffrn_measurement.diffrn_id

This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_measurement.method (*text*)

_diffrn_measurement_method (*cif_core.dic 2.0.1*)

Method used to measure intensities.

Example: 'profile data from theta/2theta scans'.

[diffrn_measurement]

_diffrn_measurement.specimen_support (*text*)
_diffrn_measurement_specimen_support (*cif_core.dic 2.0.1*)
The physical device used to support the crystal during data collection.
Examples: 'glass capillary', 'quartz capillary', 'fiber', 'metal loop'.
[diffrn_measurement]

DIFFRN_ORIENT_MATRIX

Data items in the DIFFRN_ORIENT_MATRIX category record details about the orientation matrix used in the measurement of the diffraction data.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_orient_matrix.diffrn_id**

Example 1 – based on CAD-4 diffractometer data obtained for Yb(S-C_5H_4N)$_2$-(THF)$_4$.

```
_diffrn_orient_matrix.diffrn_id   set1
_diffrn_orient_matrix.type
; reciprocal axis matrix, multiplies hkl vector to generate
  diffractometer xyz vector and diffractometer angles
;
_diffrn_orient_matrix.UB[1][1]    -0.071479
_diffrn_orient_matrix.UB[1][2]     0.020208
_diffrn_orient_matrix.UB[1][3]     0.039076
_diffrn_orient_matrix.UB[2][1]     0.035372
_diffrn_orient_matrix.UB[2][2]     0.056209
_diffrn_orient_matrix.UB[2][3]     0.078324
_diffrn_orient_matrix.UB[3][1]    -0.007470
_diffrn_orient_matrix.UB[3][2]     0.067854
_diffrn_orient_matrix.UB[3][3]    -0.017832
```

* **_diffrn_orient_matrix.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_orient_matrix.type (*text*)
_diffrn_orient_matrix_type (*cif_core.dic 2.0.1*)
A description of the orientation matrix type and how it should be applied to define the orientation of the crystal precisely with respect to the diffractometer axes.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[1][1] (*float*)
_diffrn_orient_matrix_UB_11 (*cif_core.dic 2.0.1*)
The [1][1] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[1][2] (*float*)
_diffrn_orient_matrix_UB_12 (*cif_core.dic 2.0.1*)
The [1][2] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[1][3] (*float*)
_diffrn_orient_matrix_UB_13 (*cif_core.dic 2.0.1*)
The [1][3] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[2][1] (*float*)
_diffrn_orient_matrix_UB_21 (*cif_core.dic 2.0.1*)
The [2][1] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[2][2] (*float*)
_diffrn_orient_matrix_UB_22 (*cif_core.dic 2.0.1*)
The [2][2] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[2][3] (*float*)
_diffrn_orient_matrix_UB_23 (*cif_core.dic 2.0.1*)
The [2][3] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[3][1] (*float*)
_diffrn_orient_matrix_UB_31 (*cif_core.dic 2.0.1*)
The [3][1] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[3][2] (*float*)
_diffrn_orient_matrix_UB_32 (*cif_core.dic 2.0.1*)
The [3][2] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

_diffrn_orient_matrix.UB[3][3] (*float*)
_diffrn_orient_matrix_UB_33 (*cif_core.dic 2.0.1*)
The [3][3] element of the 3 × 3 matrix that defines the dimensions of the reciprocal cell and its orientation with respect to the local diffractometer axes. See also **_diffrn_orient_matrix.type**.
[diffrn_orient_matrix]

DIFFRN_ORIENT_REFLN

Data items in the DIFFRN_ORIENT_REFLN category record details about the reflections that define the orientation matrix used in the measurement of the diffraction intensities.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_orient_refln.diffrn_id**
_diffrn_orient_refln.index_h
_diffrn_orient_refln.index_k
_diffrn_orient_refln.index_l

Example 1 – based on CAD-4 diffractometer data obtained for Yb(S-C_5H_4N)$_2$-(THF)$_4$.

```
_diffrn_orient_refln.diffrn_id    myset1
_diffrn_orient_refln.index_h      2
_diffrn_orient_refln.index_k      0
_diffrn_orient_refln.index_l      2
_diffrn_orient_refln.angle_chi    -28.45
_diffrn_orient_refln.angle_kappa  -11.32
_diffrn_orient_refln.angle_omega  5.33
_diffrn_orient_refln.angle_phi    101.78
_diffrn_orient_refln.angle_psi    0.00
_diffrn_orient_refln.angle_theta  10.66
# ... data abbreviated  ...
```

_diffrn_orient_refln.angle_chi (*float*)
_diffrn_orient_refln_angle_chi (*cif_core.dic 2.0.1*)
Diffractometer angle χ of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix.UB**[][] and the Miller indices in the DIFFRN_ORIENT_REFLN category.
[diffrn_orient_refln]

_diffrn_orient_refln.angle_kappa (*float*)
_diffrn_orient_refln_angle_kappa (*cif_core.dic 2.0.1*)
Diffractometer angle κ of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix.UB**[][] and the Miller indices in the DIFFRN_ORIENT_REFLN category.
[diffrn_orient_refln]

_diffrn_orient_refln.angle_omega (*float*)
_diffrn_orient_refln_angle_omega (*cif_core.dic 2.0.1*)
Diffractometer angle ω of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix.UB**[][] and the Miller indices in the DIFFRN_ORIENT_REFLN category.
[diffrn_orient_refln]

_diffrn_orient_refln.angle_phi (*float*)
_diffrn_orient_refln_angle_phi (*cif_core.dic 2.0.1*)
Diffractometer angle φ of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix.UB**[][] and the Miller indices in the DIFFRN_ORIENT_REFLN category.
[diffrn_orient_refln]

_diffrn_orient_refln.angle_psi (*float*)
_diffrn_orient_refln_angle_psi (*cif_core.dic 2.0.1*)
Diffractometer angle ψ of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix.UB**[][] and the Miller indices in the DIFFRN_ORIENT_REFLN category.
[diffrn_orient_refln]

_diffrn_orient_refln.angle_theta (*float*)
_diffrn_orient_refln_angle_theta (*cif_core.dic 2.0.1*)
Diffractometer angle θ of a reflection used to define the orientation matrix in degrees. See **_diffrn_orient_matrix.UB**[][] and the Miller indices in the DIFFRN_ORIENT_REFLN category.
[diffrn_orient_refln]

* **_diffrn_orient_refln.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

* **_diffrn_orient_refln.index_h** (*int*)
_diffrn_orient_refln_index_h (*cif_core.dic 2.0.1*)
Miller index *h* of a reflection used to define the orientation matrix.
[diffrn_orient_refln]

* **_diffrn_orient_refln.index_k** (*int*)
_diffrn_orient_refln_index_k (*cif_core.dic 2.0.1*)
Miller index *k* of a reflection used to define the orientation matrix.
[diffrn_orient_refln]

* **_diffrn_orient_refln.index_l** (*int*)
_diffrn_orient_refln_index_l (*cif_core.dic 2.0.1*)
Miller index *l* of a reflection used to define the orientation matrix.
[diffrn_orient_refln]

DIFFRN_RADIATION

Data items in the DIFFRN_RADIATION category describe the radiation used in measuring the diffraction intensities, its collimation and monochromatization before the sample. Post-sample treatment of the beam is described by data items in the DIFFRN_DETECTOR category.

Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_radiation.diffrn_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_radiation.diffrn_id          'set1'

_diffrn_radiation.collimation        '0.3 mm double pinhole'
_diffrn_radiation.monochromator      'graphite'
_diffrn_radiation.type               'Cu K\a'
_diffrn_radiation.wavelength_id       1
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
_diffrn_radiation.wavelength_id   1
_diffrn_radiation.type            'Cu K\a'
_diffrn_radiation.monochromator   'graphite'
```

_diffrn_radiation.collimation (*text*)
_diffrn_radiation_collimation (*cif_core.dic 2.0.1*)
The collimation or focusing applied to the radiation.
Examples: '`0.3 mm double-pinhole`', '`0.5 mm`', '`focusing mirrors`'.
[diffrn_radiation]

* **_diffrn_radiation.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_radiation.filter_edge (*float*)
_diffrn_radiation_filter_edge (*cif_core.dic 2.0.1*)
Absorption edge in ångströms of the radiation filter used.
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.inhomogeneity (*float*)
_diffrn_radiation_inhomogeneity (*cif_core.dic 2.0.1*)
Half-width in millimetres of the incident beam in the direction perpendicular to the diffraction plane.
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.monochromator (*text*)
_diffrn_radiation_monochromator (*cif_core.dic 2.0.1*)
The method used to obtain monochromatic radiation. If a monochromator crystal is used, the material and the indices of the Bragg reflection are specified.
Examples: '`Zr filter`', '`Ge 220`', '`none`', '`equatorial mounted graphite`'.
[diffrn_radiation]

_diffrn_radiation.polarisn_norm (*float*)
_diffrn_radiation_polarisn_norm (*cif_core.dic 2.0.1*)
The angle in degrees, as viewed from the specimen, between the perpendicular component of the polarization and the diffraction plane. See **_diffrn_radiation.polarisn_ratio**.
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.polarisn_ratio (*float*)
_diffrn_radiation_polarisn_ratio (*cif_core.dic 2.0.1*)
Polarization ratio of the diffraction beam incident on the crystal. This is the ratio of the perpendicularly polarized to the parallel-polarized component of the radiation. The perpendicular component forms an angle of **_diffrn_radiation.polarisn_norm** to the normal to the diffraction plane of the sample (*i.e.* the plane containing the incident and reflected beams).
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.probe (*line*)

_diffrn_radiation_probe (*cif_core.dic 2.0.1*)

The nature of the radiation used (*i.e.* the name of the subatomic particle or the region of the electromagnetic spectrum). It is strongly recommended that this information be given, so that the probe radiation can be simply determined.

The data value must be one of the following:

`x-ray`
`neutron`
`electron`
`gamma`

[diffrn_radiation]

_diffrn_radiation.type (*line*)

_diffrn_radiation_type (*cif_core.dic 2.0.1*)

The nature of the radiation. This is typically a description of the X-ray wavelength in Siegbahn notation.

Examples: '`CuK\a`', '`Cu K\a~1~`', '`Cu K-L~2,3~`', '`white-beam`'.

[diffrn_radiation]

*_diffrn_radiation.wavelength_id

This data item is a pointer to **_diffrn_radiation_wavelength.id** in the DIFFRN_RADIATION_WAVELENGTH category.

_diffrn_radiation.xray_symbol (*line*)

_diffrn_radiation_xray_symbol (*cif_core.dic 2.0.1*)

The IUPAC symbol for the X-ray wavelength for the probe radiation.

The data value must be one of the following:

`K-L~3~`	$K\alpha_1$ in older Siegbahn notation
`K-L~2~`	$K\alpha_2$ in older Siegbahn notation
`K-M~3~`	$K\beta$ in older Siegbahn notation
`K-L~2,3~`	use where K-L_3 and K-L_2 are not resolved

[diffrn_radiation]

DIFFRN_RADIATION_WAVELENGTH

Data items in the DIFFRN_RADIATION_WAVELENGTH category describe the wavelength of the radiation used to measure the diffraction intensities. Items may be looped to identify and assign weights to distinct components of a polychromatic beam.

Category group(s): **inclusive_group**, **diffrn_group**
Category key(s): **_diffrn_radiation_wavelength.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_radiation_wavelength.id              1
_diffrn_radiation_wavelength.wavelength      1.54
_diffrn_radiation_wavelength.wt              1.0
```

*_diffrn_radiation_wavelength.id (*code*)

_diffrn_radiation_wavelength_id (*cif_core.dic 2.0.1*)

The code identifying each value of **_diffrn_radiation_wavelength.wavelength**. Items in the DIFFRN_RADIATION_WAVELENGTH category are looped when multiple wavelengths are used. This code is used to link with the DIFFRN_REFLN category. The **_diffrn_refln.wavelength_id** codes must match one of the codes defined in this category.

The following item(s) have an equivalent role in their respective categories:

_diffrn_radiation.wavelength_id,
_diffrn_refln.wavelength_id,
_refln.wavelength_id.

Examples: '`x1`', '`x2`', '`neut`'. **[diffrn_radiation_wavelength]**

*_diffrn_radiation_wavelength.wavelength (*float*)

_diffrn_radiation_wavelength (*cif_core.dic 2.0.1*)

The radiation wavelength in ångströms.

The permitted range is $[0.0, \infty)$. **[diffrn_radiation_wavelength]**

_diffrn_radiation_wavelength.wt (*float*)

_diffrn_radiation_wavelength_wt (*cif_core.dic 2.0.1*)

The relative weight of a wavelength identified by the code **_diffrn_radiation_wavelength.id** in the list of wavelengths.

The permitted range is [0.0, 1.0]. Where no value is given, the assumed value is '`1.0`'.

[diffrn_radiation_wavelength]

DIFFRN_REFLN

Data items in the DIFFRN_REFLN category record details about the intensities in the diffraction data set identified by **_diffrn_refln.diffrn_id**. The DIFFRN_REFLN data items refer to individual intensity measurements and must be included in looped lists. The DIFFRN_REFLNS data items specify the parameters that apply to all intensity measurements in the particular diffraction data set identified by **_diffrn_reflns.diffrn_id**.

Category group(s): **inclusive_group**, **diffrn_group**
Category key(s): **_diffrn_refln.diffrn_id**, **_diffrn_refln.id**

Example 1 – based on CAD-4 diffractometer data obtained for Yb(S-C_5H_4N)$_2$-(THF)$_4$ for data set 'set1' reflection 1102.

```
_diffrn_refln.diffrn_id             set1
_diffrn_refln.id                    1102
_diffrn_refln.wavelength_id         Cu1fixed
_diffrn_refln.angle_chi             32.21
_diffrn_refln.angle_kappa           20.12
_diffrn_refln.angle_omega           11.54
_diffrn_refln.angle_phi             176.02
_diffrn_refln.angle_psi             0.00
_diffrn_refln.angle_theta           23.08
_diffrn_refln.attenuator_code       'Ni.005'
_diffrn_refln.counts_bg_1           22
_diffrn_refln.counts_bg_2           25
_diffrn_refln.counts_net            3450
_diffrn_refln.counts_peak           321
_diffrn_refln.counts_total          3499
_diffrn_refln.detect_slit_horiz     0.04
_diffrn_refln.detect_slit_vert      0.02
_diffrn_refln.elapsed_time          1.00
_diffrn_refln.index_h               4
_diffrn_refln.index_k               0
_diffrn_refln.index_l               2
_diffrn_refln.intensity_net         202.56
_diffrn_refln.intensity_sigma       2.18
_diffrn_refln.scale_group_code      A24
_diffrn_refln.scan_mode             om
_diffrn_refln.scan_mode_backgd      mo
_diffrn_refln.scan_rate             1.2
_diffrn_refln.scan_time_backgd      900.00
_diffrn_refln.scan_width            1.0
_diffrn_refln.sint_over_lambda      0.25426
_diffrn_refln.standard_code         1
_diffrn_refln.wavelength            1.54184
```

_diffrn_refln.angle_chi (*float*)

_diffrn_refln_angle_chi (*cif_core.dic 2.0.1*)

The diffractometer angle χ of a reflection in degrees. This angle corresponds to the specified orientation matrix and the original measured cell before any subsequent cell transformations.

[diffrn_refln]

_diffrn_refln.angle_kappa (*float*)

_diffrn_refln_angle_kappa (*cif_core.dic 2.0.1*)

The diffractometer angle κ of a reflection in degrees. This angle corresponds to the specified orientation matrix and the original measured cell before any subsequent cell transformations.

[diffrn_refln]

_diffrn_refln.angle_omega (*float*)
_diffrn_refln_angle_omega (*cif_core.dic 2.0.1*)
The diffractometer angle ω of a reflection in degrees. This angle corresponds to the specified orientation matrix and the original measured cell before any subsequent cell transformations.
[diffrn_refln]

_diffrn_refln.angle_phi (*float*)
_diffrn_refln_angle_phi (*cif_core.dic 2.0.1*)
The diffractometer angle φ of a reflection in degrees. This angle corresponds to the specified orientation matrix and the original measured cell before any subsequent cell transformations.
[diffrn_refln]

_diffrn_refln.angle_psi (*float*)
_diffrn_refln_angle_psi (*cif_core.dic 2.0.1*)
The diffractometer angle ψ of a reflection in degrees. This angle corresponds to the specified orientation matrix and the original measured cell before any subsequent cell transformations.
[diffrn_refln]

_diffrn_refln.angle_theta (*float*)
_diffrn_refln_angle_theta (*cif_core.dic 2.0.1*)
The diffractometer angle θ of a reflection in degrees. This angle corresponds to the specified orientation matrix and the original measured cell before any subsequent cell transformations.
[diffrn_refln]

_diffrn_refln.attenuator_code
_diffrn_refln_attenuator_code (*cif_core.dic 2.0.1*)
The code identifying the attenuator setting for this reflection. This code must match one of the **_diffrn_attenuator.code** values.

_diffrn_refln.class_code (*code*)
_diffrn_refln_class_code (*cif_core.dic 2.3*)
The code identifying the class to which this reflection has been assigned. This code must match a value of **_diffrn_reflns.class_code**. Reflections may be grouped into classes for a variety of purposes. For example, for modulated structures each reflection class may be defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal lattice.
[diffrn_refln]

_diffrn_refln.counts_bg_1 (*int*)
_diffrn_refln_counts_bg_1 (*cif_core.dic 2.0.1*)
The diffractometer counts for the measurement of the background before the peak.
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.counts_bg_2 (*int*)
_diffrn_refln_counts_bg_2 (*cif_core.dic 2.0.1*)
The diffractometer counts for the measurement of the background after the peak.
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.counts_net (*int*)
_diffrn_refln_counts_net (*cif_core.dic 2.0.1*)
The diffractometer counts for the measurement of net counts after background removal.
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.counts_peak (*int*)
_diffrn_refln_counts_peak (*cif_core.dic 2.0.1*)
The diffractometer counts for the measurement of counts for the peak scan or position.
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.counts_total (*int*)
_diffrn_refln_counts_total (*cif_core.dic 2.0.1*)
The diffractometer counts for the measurement of total counts (background plus peak).
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.detect_slit_horiz (*float*)
_diffrn_refln_detect_slit_horiz (*cif_core.dic 2.0.1*)
Total slit aperture in degrees in the diffraction plane.
The permitted range is $[0.0, 90.0]$. **[diffrn_refln]**

_diffrn_refln.detect_slit_vert (*float*)
_diffrn_refln_detect_slit_vert (*cif_core.dic 2.0.1*)
Total slit aperture in degrees perpendicular to the diffraction plane.
The permitted range is $[0.0, 90.0]$. **[diffrn_refln]**

* **_diffrn_refln.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_refln.elapsed_time (*float*)
_diffrn_refln_elapsed_time (*cif_core.dic 2.0.1*)
Elapsed time in minutes from the start of the diffraction experiment to the measurement of this intensity.
The permitted range is $[0.0, \infty)$. **[diffrn_refln]**

* **_diffrn_refln.id** (*code*)
The value of **_diffrn_refln.id** must uniquely identify the reflection in the data set identified by the item **_diffrn_refln.diffrn_id**. Note that this item need not be a number; it can be any unique identifier.
[diffrn_refln]

* **_diffrn_refln.index_h** (*int*)
_diffrn_refln_index_h (*cif_core.dic 2.0.1*)
Miller index h of a reflection. The values of the Miller indices in the DIFFRN_REFLN category need not match the values of the Miller indices in the REFLN category if a transformation of the original measured cell has taken place. Details of the cell transformation are given in **_diffrn_reflns.reduction_process**. See also **_diffrn_reflns.transf_matrix**[][].
[diffrn_refln]

* **_diffrn_refln.index_k** (*int*)
_diffrn_refln_index_k (*cif_core.dic 2.0.1*)
Miller index k of a reflection. The values of the Miller indices in the DIFFRN_REFLN category need not match the values of the Miller indices in the REFLN category if a transformation of the original measured cell has taken place. Details of the cell transformation are given in **_diffrn_reflns.reduction_process**. See also **_diffrn_reflns.transf_matrix**[][].
[diffrn_refln]

* **_diffrn_refln.index_l** (*int*)
_diffrn_refln_index_l (*cif_core.dic 2.0.1*)
Miller index l of a reflection. The values of the Miller indices in the DIFFRN_REFLN category need not match the values of the Miller indices in the REFLN category if a transformation of the original measured cell has taken place. Details of the cell transformation are given in **_diffrn_reflns.reduction_process**. See also **_diffrn_reflns.transf_matrix**[][].
[diffrn_refln]

_diffrn_refln.intensity_net (*float*)
_diffrn_refln_intensity_net (*cif_core.dic 2.0.1*)
Net intensity calculated from the diffraction counts after the attenuator and standard scales have been applied.
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.intensity_sigma (*float*)
_diffrn_refln_intensity_sigma (*cif_core.dic 2.0.1*)
Standard uncertainty (estimated standard deviation) of the intensity calculated from the diffraction counts after the attenuator and standard scales have been applied.
The permitted range is $[0, \infty)$. **[diffrn_refln]**

_diffrn_refln.intensity_u (*float*)
_diffrn_refln_intensity_u (*cif_core.dic 2.3*)
Standard uncertainty of the net intensity calculated from the diffraction counts after the attenuator and standard scales have been applied.
The permitted range is $[0.0, \infty)$.
Related item: **_diffrn_refln.intensity_sigma** (alternate). **[diffrn_refln]**

***_diffrn_refln.scale_group_code**
_diffrn_refln_scale_group_code (*cif_core.dic 2.0.1*)
The code identifying the scale applying to this reflection. This data item is a pointer to **_diffrn_scale_group.code** in the DIFFRN_SCALE_GROUP category.

_diffrn_refln.scan_mode (*ucode*)
_diffrn_refln_scan_mode (*cif_core.dic 2.0.1*)
The code identifying the mode of scanning for measurements using a diffractometer. See **_diffrn_refln.scan_width** and **_diffrn_refln.scan_mode_backgd**.
The data value must be one of the following:

om	ω scan
ot	$\omega/2\theta$ scan
q	Q scans (arbitrary reciprocal directions)

[diffrn_refln]

_diffrn_refln.scan_mode_backgd (*ucode*)
_diffrn_refln_scan_mode_backgd (*cif_core.dic 2.0.1*)
The code identifying the mode of scanning a reflection to measure the background intensity.
The data value must be one of the following:

st	stationary counter background
mo	moving counter background

[diffrn_refln]

_diffrn_refln.scan_rate (*float*)
_diffrn_refln_scan_rate (*cif_core.dic 2.0.1*)
The rate of scanning a reflection in degrees per minute to measure the intensity.
[diffrn_refln]

_diffrn_refln.scan_time_backgd (*float*)
_diffrn_refln_scan_time_backgd (*cif_core.dic 2.0.1*)
The time spent measuring each background in seconds.
[diffrn_refln]

_diffrn_refln.scan_width (*float*)
_diffrn_refln_scan_width (*cif_core.dic 2.0.1*)
The scan width in degrees of the scan mode defined by the code **_diffrn_refln.scan_mode**.
The permitted range is $[0.0, 90.0]$. **[diffrn_refln]**

_diffrn_refln.sint_over_lambda (*float*)
_diffrn_refln_sint/lambda (*cif_core.dic 2.0.1*)
The $(\sin\theta)/\lambda$ value in reciprocal ångströms for this reflection.
The permitted range is $[0.0, \infty)$. **[diffrn_refln]**

***_diffrn_refln.standard_code**
_diffrn_refln_standard_code (*cif_core.dic 2.0.1*)
The code identifying that this reflection was measured as a standard intensity. This data item is a pointer to **_diffrn_standard_refln.code** in the DIFFRN_STANDARD_REFLN category.

_diffrn_refln.wavelength (*float*)
_diffrn_refln_wavelength (*cif_core.dic 2.0.1*)
The mean wavelength in ångströms of the radiation used to measure the intensity of this reflection. This is an important parameter for data collected using energy-dispersive detectors or the Laue method.
The permitted range is $[0.0, \infty)$. **[diffrn_refln]**

***_diffrn_refln.wavelength_id**
_diffrn_refln_wavelength_id (*cif_core.dic 2.0.1*)
This data item is a pointer to **_diffrn_radiation.wavelength_id** in the DIFFRN_RADIATION category.

DIFFRN_REFLNS

Data items in the DIFFRN_REFLNS category record details about the set of intensities measured in the diffraction experiment. The DIFFRN_REFLN data items refer to individual intensity measurements and must be included in looped lists. The DIFFRN_REFLNS data items specify the parameters that apply to all intensity measurements in a diffraction data set.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_reflns.diffrn_id**

_diffrn_reflns.av_R_equivalents (*float*)
_diffrn_reflns_av_R_equivalents (*cif_core.dic 2.0.1*)
The residual $[\sum \mathrm{av}|\Delta(I)| / \sum |\mathrm{av}(I)|]$ for symmetry-equivalent reflections used to calculate the average intensity $\mathrm{av}(I)$. The $\mathrm{av}|\Delta(I)|$ term is the average absolute difference between $\mathrm{av}(I)$ and the individual symmetry-equivalent intensities.
The permitted range is $[0.0, \infty)$. **[diffrn_reflns]**

_diffrn_reflns.av_sigmaI_over_netI (*float*)
_diffrn_reflns_av_sigmaI/netI (*cif_core.dic 2.0.1*)
Measure $[\sum |\sigma(\mathrm{net}\ I)| / \sum |\mathrm{net}\ I|]$ for all measured reflections.
The permitted range is $[0.0, \infty)$. **[diffrn_reflns]**

_diffrn_reflns.av_unetI/netI (*float*)
_diffrn_reflns_av_unetI/netI (*cif_core.dic 2.3*)
Measure $[\sum |u(\mathrm{net}\ I)| / \sum |\mathrm{net}\ I|]$ for all measured reflections.
The permitted range is $[0.0, \infty)$. **[diffrn_reflns]**

***_diffrn_reflns.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_reflns.limit_h_max (*int*)
_diffrn_reflns_limit_h_max (*cif_core.dic 2.0.1*)
The maximum value of the Miller index h for the reflection data specified by **_diffrn_refln.index_h**.
[diffrn_reflns]

_diffrn_reflns.limit_h_min (*int*)
_diffrn_reflns_limit_h_min (*cif_core.dic 2.0.1*)
The minimum value of the Miller index h for the reflection data specified by **_diffrn_refln.index_h**.
[diffrn_reflns]

_diffrn_reflns.limit_k_max (*int*)
_diffrn_reflns_limit_k_max (*cif_core.dic 2.0.1*)
The maximum value of the Miller index k for the reflection data specified by **_diffrn_refln.index_k**.
[diffrn_reflns]

_diffrn_reflns.limit_k_min (*int*)
_diffrn_reflns_limit_k_min (*cif_core.dic 2.0.1*)
The minimum value of the Miller index k for the reflection data specified by **_diffrn_refln.index_k**.
[diffrn_reflns]

_diffrn_reflns.limit_l_max (*int*)
_diffrn_reflns_limit_l_max (*cif_core.dic 2.0.1*)
The maximum value of the Miller index l for the reflection data specified by **_diffrn_refln.index_l**.
[diffrn_reflns]

_diffrn_reflns.limit_l_min (*int*)
_diffrn_reflns_limit_l_min (*cif_core.dic 2.0.1*)
The minimum value of the Miller index l for the reflection data specified by **_diffrn_refln.index_l**.
[diffrn_reflns]

_diffrn_reflns.number (*int*)
_diffrn_reflns_number (*cif_core.dic 2.0.1*)
The total number of measured intensities, excluding reflections that are classified as systematically absent.
The permitted range is $[0, \infty)$. **[diffrn_reflns]**

_diffrn_reflns.reduction_process (*text*)
_diffrn_reflns_reduction_process (*cif_core.dic 2.0.1*)
A description of the process used to reduce the intensity data into structure-factor magnitudes.
Example: 'data averaged using Fisher test'. **[diffrn_reflns]**

_diffrn_reflns.theta_max (*float*)
_diffrn_reflns_theta_max (*cif_core.dic 2.0.1*)
Maximum θ angle in degrees for the measured diffraction intensities.
The permitted range is [0.0, 90.0]. **[diffrn_reflns]**

_diffrn_reflns.theta_min (*float*)
_diffrn_reflns_theta_min (*cif_core.dic 2.0.1*)
Minimum θ angle in degrees for the measured diffraction intensities.
The permitted range is [0.0, 90.0]. **[diffrn_reflns]**

_diffrn_reflns.transf_matrix[1][1] (*float*)
_diffrn_reflns_transf_matrix_11 (*cif_core.dic 2.0.1*)
The [1][1] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[1][2] (*float*)
_diffrn_reflns_transf_matrix_12 (*cif_core.dic 2.0.1*)
The [1][2] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[1][3] (*float*)
_diffrn_reflns_transf_matrix_13 (*cif_core.dic 2.0.1*)
The [1][3] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[2][1] (*float*)
_diffrn_reflns_transf_matrix_21 (*cif_core.dic 2.0.1*)
The [2][1] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[2][2] (*float*)
_diffrn_reflns_transf_matrix_22 (*cif_core.dic 2.0.1*)
The [2][2] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[2][3] (*float*)
_diffrn_reflns_transf_matrix_23 (*cif_core.dic 2.0.1*)
The [2][3] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[3][1] (*float*)
_diffrn_reflns_transf_matrix_31 (*cif_core.dic 2.0.1*)
The [3][1] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[3][2] (*float*)
_diffrn_reflns_transf_matrix_32 (*cif_core.dic 2.0.1*)
The [3][2] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

_diffrn_reflns.transf_matrix[3][3] (*float*)
_diffrn_reflns_transf_matrix_33 (*cif_core.dic 2.0.1*)
The [3][3] element of the 3 × 3 matrix used to transform Miller indices in the DIFFRN_REFLN category into the Miller indices in the REFLN category.
[diffrn_reflns]

DIFFRN_REFLNS_CLASS

Data items in the DIFFRN_REFLNS_CLASS category record details about the classes of reflections measured in the diffraction experiment.

Category key(s): **_diffrn_reflns_class.code**

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4. Each reflection class is defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal lattice.

```
loop_
     _diffrn_reflns_class.number
     _diffrn_reflns_class.d_res_high
     _diffrn_reflns_class.d_res_low
     _diffrn_reflns_class.av_R_eq
     _diffrn_reflns_class.code
     _diffrn_reflns_class.description
  1580 0.551 6.136 0.015 'Main' 'm=0; main reflections'
  1045 0.551 6.136 0.010 'Sat1' 'm=1; first-order satellites'
```

_diffrn_reflns_class.av_R_eq (*float*)

_diffrn_reflns_class_av_R_eq (*cif_core.dic 2.3*)

For each reflection class, the residual $[\sum \mathrm{av}|\Delta(I)| / \sum |\mathrm{av}(I)|]$ for symmetry-equivalent reflections used to calculate the average intensity $\mathrm{av}(I)$. The $\mathrm{av}|\Delta(I)|$ term is the average absolute difference between $\mathrm{av}(I)$ and the individual intensities.

The permitted range is $[0.0, \infty)$. **[diffrn_reflns_class]**

_diffrn_reflns_class.av_sgI/I (*float*)

_diffrn_reflns_class_av_sgI/I (*cif_core.dic 2.3*)

Measure $[\sum |\sigma(\mathrm{net}\ I)| / \sum |\mathrm{net}\ I|]$ for all measured intensities in a reflection class.

The permitted range is $[0.0, \infty)$.

Related item: **_diffrn_reflns_class.av_uI/I** (replaces).

[diffrn_reflns_class]

_diffrn_reflns_class.av_uI/I (*float*)

_diffrn_reflns_class_av_uI/I (*cif_core.dic 2.3*)

Measure $[\sum |u(\mathrm{net}\ I)| / \sum |\mathrm{net}\ I|]$ for all measured intensities in a reflection class.

The permitted range is $[0.0, \infty)$.

Related item: **_diffrn_reflns_class.av_sgI/I** (alternate).

[diffrn_reflns_class]

*_diffrn_reflns_class.code (*code*)

_diffrn_reflns_class_code (*cif_core.dic 2.3*)

The code identifying a certain reflection class.

Examples: '1', 'm1', 's2'. **[diffrn_reflns_class]**

_diffrn_reflns_class.d_res_high (*float*)

_diffrn_reflns_class_d_res_high (*cif_core.dic 2.3*)

The smallest value in ångströms for the interplanar spacings for the reflections in each measured reflection class. This is called the highest resolution for this reflection class.

The permitted range is $[0.0, \infty)$. **[diffrn_reflns_class]**

_diffrn_reflns_class.d_res_low (*float*)

_diffrn_reflns_class_d_res_low (*cif_core.dic 2.3*)

The largest value in ångströms of the interplanar spacings for the reflections for each measured reflection class. This is called the lowest resolution for this reflection class.

The permitted range is $[0.0, \infty)$. **[diffrn_reflns_class]**

_diffrn_reflns_class.description (*text*)

_diffrn_reflns_class_description (*cif_core.dic 2.3*)

Description of each reflection class.

Examples: 'm=1 first order satellites',
'H0L0 common projection reflections'. **[diffrn_reflns_class]**

_diffrn_reflns_class.number (*int*)

_diffrn_reflns_class_number (*cif_core.dic 2.3*)

The total number of measured intensities for each reflection class, excluding the systematic absences arising from centring translations.

The permitted range is $[0, \infty)$. **[diffrn_reflns_class]**

DIFFRN_SCALE_GROUP

Data items in the DIFFRN_SCALE_GROUP category record details of the scaling factors applied to place all intensities in the reflection lists on a common scale. Scaling groups might, for example, correspond to each film in a multi-film data set or each crystal in a multi-crystal data set.

Category group(s): **inclusive_group**
diffrn_group

Category key(s): **_diffrn_scale_group.code**

Example 1 – based on CAD-4 diffractometer data obtained for Yb(S-$C_5H_4N)_2$-$(THF)_4$.

```
_diffrn_scale_group.code       A24
_diffrn_scale_group.I_net      1.021
```

*_diffrn_scale_group.code (*code*)

_diffrn_scale_group_code (*cif_core.dic 2.0.1*)

The value of **_diffrn_scale_group.code** must uniquely identify a record in the DIFFRN_SCALE_GROUP list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_diffrn_refln.scale_group_code.

Examples: '1', '2', 'c1', 'c2'. **[diffrn_scale_group]**

_diffrn_scale_group.I_net (*float*)

_diffrn_scale_group_I_net (*cif_core.dic 2.0.1*)

The scale for a specific measurement group which is to be multiplied with the net intensity to place all intensities in the DIFFRN_REFLN or REFLN list on a common scale.

The permitted range is $[0.0, \infty)$. **[diffrn_scale_group]**

DIFFRN_SOURCE

Data items in the DIFFRN_SOURCE category record details of the source of radiation used in the diffraction experiment.

Category group(s): **inclusive_group**
diffrn_group

Category key(s): **_diffrn_source.diffrn_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_source.diffrn_id      's1'
_diffrn_source.source         'rotating anode'
_diffrn_source.type           'Rigaku RU-200'
_diffrn_source.power          50
_diffrn_source.current        180
_diffrn_source.size           '8mm x 0.4 mm broad-focus'
```

_diffrn_source.current (*float*)

_diffrn_source_current (*cif_core.dic 2.0.1*)

The current in milliamperes at which the radiation source was operated.

[diffrn_source]

_diffrn_source.details (*text*)
_diffrn_source_details (*cif_core.dic 2.0.1*)
A description of special aspects of the radiation source used.
[diffrn_source]

* **_diffrn_source.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_source.power (*float*)
_diffrn_source_power (*cif_core.dic 2.0.1*)
The power in kilowatts at which the radiation source was operated.
[diffrn_source]

_diffrn_source.size (*text*)
_diffrn_source_size (*cif_core.dic 2.0.1*)
The dimensions of the source as viewed from the sample.
Examples: '8mm x 0.4 mm fine-focus', 'broad focus'. **[diffrn_source]**

_diffrn_source.source (*text*)
_diffrn_radiation_source (*cifdic.c91 1.0*)
_diffrn_source (*cif_core.dic 2.0*)
The general class of the radiation source.
Examples: 'sealed X-ray tube', 'nuclear reactor', 'spallation source', 'electron microscope', 'rotating-anode X-ray tube', 'synchrotron'.
[diffrn_source]

_diffrn_source.take-off_angle (*float*)
_diffrn_source_take-off_angle (*cif_core.dic 2.3*)
The complement of the angle in degrees between the normal to the surface of the X-ray tube target and the primary X-ray beam for beams generated by traditional X-ray tubes.
The permitted range is [0.00, 90.0].
Example: '1.5'. **[diffrn_source]**

_diffrn_source.target (*code*)
_diffrn_source_target (*cif_core.dic 2.0.1*)
The chemical element symbol for the X-ray target (usually the anode) used to generate X-rays. This can also be used for spallation sources.
The data value must be one of the following:

H	He	Li	Be	B	C	N	O	F	Ne	Na	Mg	Al	Si	P
S	Cl	Ar	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn
Ga	Ge	As	Se	Br	Kr	Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh
Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe	Cs	Ba	La	Ce	Pr	Nd
Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb	Lu	Hf	Ta	W	Re
Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Rn	Fr	Ra	Ac	Th
Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No	Lr		

[diffrn_source]

_diffrn_source.type (*text*)
_diffrn_source_type (*cif_core.dic 2.0.1*)
The make, model or name of the source of radiation.
Examples: 'NSLS beamline X8C', 'Rigaku RU200'. **[diffrn_source]**

_diffrn_source.voltage (*float*)
_diffrn_source_voltage (*cif_core.dic 2.0.1*)
The voltage in kilovolts at which the radiation source was operated.
[diffrn_source]

DIFFRN_STANDARD_REFLN

Data items in the DIFFRN_STANDARD_REFLN category record details about the reflections treated as standards during the measurement of a set of diffraction intensities. Note that these are the individual standard reflections, not the results of the analysis of the standard reflections.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_standard_refln.diffrn_id**
_diffrn_standard_refln.code

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
loop_
_diffrn_standard_refln.diffrn_id
_diffrn_standard_refln.code
_diffrn_standard_refln.index_h
_diffrn_standard_refln.index_k
_diffrn_standard_refln.index_l
 s1 1  3  2  4
 s1 1  1  9  1
 s1 1  3  0 10
```

* **_diffrn_standard_refln.code** (*code*)
_diffrn_standard_refln_code (*cif_core.dic 2.0.1*)
The code identifying a reflection measured as a standard reflection with the indices **_diffrn_standard_refln.index_h**, **_diffrn_standard_refln.index_k** and **_diffrn_standard_refln.index_l**. This is the same code as the **_diffrn_refln.standard_code** in the DIFFRN_REFLN list.
The following item(s) have an equivalent role in their respective categories:
_diffrn_refln.standard_code.
Examples: '1', '2', 'c1', 'c2'. **[diffrn_standard_refln]**

* **_diffrn_standard_refln.diffrn_id**
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

* **_diffrn_standard_refln.index_h** (*int*)
_diffrn_standard_refln_index_h (*cif_core.dic 2.0.1*)
Miller index *h* of a standard reflection used in the diffraction measurement process.
[diffrn_standard_refln]

* **_diffrn_standard_refln.index_k** (*int*)
_diffrn_standard_refln_index_k (*cif_core.dic 2.0.1*)
Miller index *k* of a standard reflection used in the diffraction measurement process.
[diffrn_standard_refln]

* **_diffrn_standard_refln.index_l** (*int*)
_diffrn_standard_refln_index_l (*cif_core.dic 2.0.1*)
Miller index *l* of a standard reflection used in the diffraction measurement process.
[diffrn_standard_refln]

DIFFRN_STANDARDS

Data items in the DIFFRN_STANDARDS category record details about the set of standard reflections used to monitor intensity stability during the measurement of diffraction intensities. Note that these records describe properties common to the set of standard reflections, not the standard reflections themselves.

Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_standards.diffrn_id**

Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_diffrn_standards.diffrn_id        's1'
_diffrn_standards.number           3
_diffrn_standards.interval_time    120
_diffrn_standards.decay_%          0
```

_diffrn_standards.decay_% (*float*)

_diffrn_standards_decay_% (*cif_core.dic 2.0.1*)

The percentage decrease in the mean of the intensities for the set of standard reflections from the start of the measurement process to the end. This value usually affords a measure of the overall decay in crystal quality during the diffraction measurement process. Negative values are used in exceptional instances where the final intensities are greater than the initial ones.

The permitted range is $(-\infty, 100.0]$. **[diffrn_standards]**

*_diffrn_standards.diffrn_id

This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_standards.interval_count (*int*)

_diffrn_standards_interval_count (*cif_core.dic 2.0.1*)

The number of reflection intensities between the measurement of standard reflection intensities.

The permitted range is $[0, \infty)$. **[diffrn_standards]**

_diffrn_standards.interval_time (*float*)

_diffrn_standards_interval_time (*cif_core.dic 2.0.1*)

The time in minutes between the measurement of standard reflection intensities.

The permitted range is $[0, \infty)$. **[diffrn_standards]**

_diffrn_standards.number (*int*)

_diffrn_standards_number (*cif_core.dic 2.0.1*)

The number of unique standard reflections used during the measurement of the diffraction intensities.

The permitted range is $[0, \infty)$. **[diffrn_standards]**

_diffrn_standards.scale_sigma (*float*)

_diffrn_standards_scale_sigma (*cif_core.dic 2.0.1*)

The standard uncertainty (estimated standard deviation) of the individual mean standard scales applied to the intensity data.

The permitted range is $[0.0, \infty)$. **[diffrn_standards]**

_diffrn_standards.scale_u (*float*)

_diffrn_standards_scale_u (*cif_core.dic 2.3*)

The standard uncertainty of the individual mean standard scales applied to the intensity data.

The permitted range is $[0.0, \infty)$.

Related item: **_diffrn_standards.scale_sigma** (alternate).

[diffrn_standards]

ENTITY

Data items in the ENTITY category record details (such as chemical composition, name and source) about the molecular entities that are present in the crystallographic structure. Items in the various ENTITY subcategories provide a full chemical description of these molecular entities. Entities are of three types: polymer, non-polymer and water. Note that the water category includes only water; ordered solvent such as sulfate ion or acetone would be described as individual non-polymer entities. The ENTITY category is specific to macromolecular CIF applications and replaces the function of the CHEMICAL category in the CIF core. It is important to remember that the ENTITY data are not the result of the crystallographic experiment; those results are represented by the ATOM_SITE data items. ENTITY data items describe the chemistry of the molecules under investigation and can most usefully be thought of as the ideal groups to which the structure is restrained or constrained during refinement. It is also important to remember that entities do not correspond directly to the enumeration of the contents of the asymmetric unit. Entities are described only once, even in those structures that contain multiple observations of an entity. The STRUCT_ASYM data items, which reference the entity list, describe and label the contents of the asymmetric unit.

Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity.id
_entity.type
_entity.formula_weight
_entity.details
 1 polymer      10916
;  The enzymatically competent form of HIV protease is a
   dimer. This entity corresponds to one monomer of an
   active dimer.
;
 2 non-polymer 647.2  .
 3 water        18    .
```

_entity.details (*text*)

A description of special aspects of the entity.

[entity]

_entity.formula_weight (*float*)

Formula mass in daltons of the entity.

The permitted range is $[1.0, \infty)$. **[entity]**

*_entity.id (*code*)

The value of **_entity.id** must uniquely identify a record in the ENTITY list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_atom_site.label_entity_id,
_entity_keywords.entity_id,
_entity_link.entity_id_1,
_entity_link.entity_id_2,
_entity_name_com.entity_id,
_entity_name_sys.entity_id,
_entity_poly.entity_id,
_entity_poly_seq.entity_id,
_entity_src_gen.entity_id,
_entity_src_nat.entity_id,
_struct_asym.entity_id,
_struct_ref.entity_id. **[entity]**

_entity.src_method (*ucode*)
The method by which the sample for the entity was produced. Entities isolated directly from natural sources (tissues, soil samples *etc.*) are expected to have further information in the ENTITY_SRC_NAT category. Entities isolated from genetically manipulated sources are expected to have further information in the ENTITY_SRC_GEN category.
The data value must be one of the following:

nat	entity isolated from a natural source
man	entity isolated from a genetically manipulated source
syn	entity obtained synthetically

[entity]

_entity.type (*ucode*)
Defines the type of the entity. Polymer entities are expected to have corresponding ENTITY_POLY and associated entries. Non-polymer entities are expected to have corresponding CHEM_COMP and associated entries. Water entities are not expected to have corresponding entries in the ENTITY category.
The data value must be one of the following:

polymer	entity is a polymer
non-polymer	entity is not a polymer
water	water in the solvent model

[entity]

ENTITY_KEYWORDS

Data items in the ENTITY_KEYWORDS category specify keywords relevant to the molecular entities. Note that this list of keywords is separate from the list that is used for the STRUCT_BIOL data items and is intended to provide only the information that one would know about the molecular entity *if one did not know its structure*. Hence polypeptides are simply polypeptides, not cytokines or β-α-barrels, and polyribonucleic acids are simply poly-RNA, not transfer-RNA.
Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_keywords.entity_id**
_entity_keywords.text

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_keywords.entity_id
_entity_keywords.text
  1  'polypeptide'
  2  'natural product, inhibitor, reduced peptide'
```

*__entity_keywords.entity_id__
This data item is a pointer to **_entity.id** in the ENTITY category.

_entity_keywords.text (*text*)
Keywords describing this entity.
Examples: 'polypeptide', 'natural product', 'polysaccharide'.
[entity_keywords]

ENTITY_LINK

Data items in the ENTITY_LINK category give details about the links between entities.
Category group(s): **inclusive_group**
chem_link_group
Category key(s): **_entity_link.link_id**

_entity_link.details (*text*)
A description of special aspects of a link between chemical components in the structure.
[entity_link]

*__entity_link.entity_id_1__
The entity ID of the first of the two entities joined by the link. This data item is a pointer to **_entity.id** in the ENTITY category.

*__entity_link.entity_id_2__
The entity ID of the second of the two entities joined by the link. This data item is a pointer to **_entity.id** in the ENTITY category.

_entity_link.entity_seq_num_1
For a polymer entity, the sequence number in the first of the two entities containing the link. This data item is a pointer to **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category.

_entity_link.entity_seq_num_2
For a polymer entity, the sequence number in the second of the two entities containing the link. This data item is a pointer to **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category.

*__entity_link.link_id__
This data item is a pointer to **_chem_link.id** in the CHEM_LINK category.

ENTITY_NAME_COM

Data items in the ENTITY_NAME_COM category record the common name or names associated with the entity. In some cases, the entity name may not be the same as the name of the biological structure. For example, haemoglobin α chain would be the entity common name, not haemoglobin.
Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_name_com.entity_id**
_entity_name_com.name

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_name_com.entity_id
_entity_name_com.name
 1  'HIV-1 protease monomer'
 1  'HIV-1 PR monomer'
 2  'acetyl-pepstatin'
 2  'acetyl-Ile-Val-Asp-Statine-Ala-Ile-Statine'
 3  'water'
```

*__entity_name_com.entity_id__
This data item is a pointer to **_entity.id** in the ENTITY category.

*__entity_name_com.name__ (*text*)
A common name for the entity.
Examples: 'HIV protease monomer', 'hemoglobin alpha chain', '2-fluoro-1,4-dichloro benzene', 'arbutin'. **[entity_name_com]**

ENTITY_NAME_SYS

Data items in the ENTITY_NAME_SYS category record the systematic name or names associated with the entity and the system that was used to construct the systematic name. In some cases, the entity name may not be the same as the name of the biological structure.

Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_name_sys.entity_id**
_entity_name_sys.name

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_name_sys.entity_id
_entity_name_sys.name
  1  'EC 3.4.23.16'
  2  'acetyl-Ile-Val-Asp-Sta-Ala-Ile-Sta'
  3  water
```

* **_entity_name_sys.entity_id**
This data item is a pointer to **_entity.id** in the ENTITY category.

* **_entity_name_sys.name** (*text*)
The systematic name for the entity.
Examples: 'hydroquinone-beta-D-pyranoside', 'EC 2.1.1.1', '2-fluoro-1,4-dichlorobenzene'. **[entity_name_sys]**

_entity_name_sys.system (*text*)
The system used to generate the systematic name of the entity.
Examples: 'Chemical Abstracts conventions', 'enzyme convention', 'Sigma catalog'. **[entity_name_sys]**

ENTITY_POLY

Data items in the ENTITY_POLY category record details about the polymer, such as the type of the polymer, the number of monomers and whether it has nonstandard features.

Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_poly.entity_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_poly.entity_id
_entity_poly.type
_entity_poly.nstd_chirality
_entity_poly.nstd_linkage
_entity_poly.nstd_monomer
_entity_poly.type_details
 1  polypeptide(L)  no  no  no  .
```

* **_entity_poly.entity_id**
This data item is a pointer to **_entity.id** in the ENTITY category.

_entity_poly.nstd_chirality (*ucode*)
A flag to indicate whether the polymer contains at least one monomer unit with chirality different from that specified in **_entity_poly.type**.
The data value must be one of the following:

no	polymer contains no monomers with different chirality
n	abbreviation for 'no'
yes	polymer contains at least one monomer with different chirality
y	abbreviation for 'yes'

[entity_poly]

_entity_poly.nstd_linkage (*ucode*)
A flag to indicate whether the polymer contains at least one monomer-to-monomer link different from that implied by **_entity_poly.type**.
The data value must be one of the following:

no	polymer contains no different links
n	abbreviation for 'no'
yes	polymer contains at least one different link
y	abbreviation for 'yes'

[entity_poly]

_entity_poly.nstd_monomer (*ucode*)
A flag to indicate whether the polymer contains at least one monomer that is not considered standard.
The data value must be one of the following:

no	polymer contains no nonstandard monomers
n	abbreviation for 'no'
yes	polymer contains at least one nonstandard monomer
y	abbreviation for 'yes'

[entity_poly]

_entity_poly.number_of_monomers (*int*)
The number of monomers in the polymer.
The permitted range is $[1, \infty)$. **[entity_poly]**

_entity_poly.type (*ucode*)
The type of the polymer.
The data value must be one of the following:

polypeptide(D)
polypeptide(L)
polydeoxyribonucleotide
polyribonucleotide
polysaccharide(D)
polysaccharide(L)
other

[entity_poly]

_entity_poly.type_details (*text*)
A description of special aspects of the polymer type.
Examples: 'monomer Ala 16 is a D-amino acid', 'the oligomer contains alternating RNA and DNA units'.
[entity_poly]

ENTITY_POLY_SEQ

Data items in the ENTITY_POLY_SEQ category specify the sequence of monomers in a polymer. Allowance is made for the possibility of microheterogeneity in a sample by allowing a given sequence number to be correlated with more than one monomer ID. The corresponding ATOM_SITE entries should reflect this heterogeneity.

Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_poly_seq.entity_id**
_entity_poly_seq.num
_entity_poly_seq.mon_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_poly_seq.entity_id
_entity_poly_seq.num
_entity_poly_seq.mon_id
  1  1  PRO   1  2  GLN   1  3  ILE   1  4  THR   1  5  LEU
  1  6  TRP   1  7  GLN   1  8  ARG   1  9  PRO   1 10  LEU
  1 11  VAL   1 12  THR   1 13  ILE   1 14  LYS   1 15  ILE
  1 16  GLY   1 17  GLY   1 18  GLN   1 19  LEU   1 20  LYS
  1 21  GLU   1 22  ALA   1 23  LEU   1 24  LEU   1 25  ASP
# - - - - data truncated for brevity - - - -
```

*__entity_poly_seq.entity_id__
This data item is a pointer to __entity.id__ in the ENTITY category.

__entity_poly_seq.hetero__ (*ucode*)
A flag to indicate whether this monomer in the polymer is heterogeneous in sequence. This would be rare.
The data value must be one of the following:

no	sequence is not heterogeneous at this monomer
n	abbreviation for 'no'
yes	sequence is heterogeneous at this monomer
y	abbreviation for 'yes'

Where no value is given, the assumed value is 'no'. **[entity_poly_seq]**

*__entity_poly_seq.mon_id__
This data item is a pointer to __chem_comp.id__ in the CHEM_COMP category.

*__entity_poly_seq.num__ (*int*)
The value of __entity_poly_seq.num__ must uniquely and sequentially identify a record in the ENTITY_POLY_SEQ list. Note that this item must be a number and that the sequence numbers must progress in increasing numerical order.
The following item(s) have an equivalent role in their respective categories:
__atom_site.label_seq_id__,
__entity_link.entity_seq_num_1__,
__entity_link.entity_seq_num_2__,
__geom_angle.atom_site_label_seq_id_1__,
__geom_angle.atom_site_label_seq_id_2__,
__geom_angle.atom_site_label_seq_id_3__,
__geom_bond.atom_site_label_seq_id_1__,
__geom_bond.atom_site_label_seq_id_2__,
__geom_contact.atom_site_label_seq_id_1__,
__geom_contact.atom_site_label_seq_id_2__,
__geom_hbond.atom_site_label_seq_id_A__,
__geom_hbond.atom_site_label_seq_id_D__,
__geom_hbond.atom_site_label_seq_id_H__,
__geom_torsion.atom_site_label_seq_id_1__,
__geom_torsion.atom_site_label_seq_id_2__,
__geom_torsion.atom_site_label_seq_id_3__,
__geom_torsion.atom_site_label_seq_id_4__,
__struct_conf.beg_label_seq_id__,
__struct_conf.end_label_seq_id__,
__struct_conn.ptnr1_label_seq_id__,
__struct_conn.ptnr2_label_seq_id__,
__struct_mon_nucl.label_seq_id__,
__struct_mon_prot.label_seq_id__,
__struct_mon_prot_cis.label_seq_id__,
__struct_ncs_dom_lim.beg_label_seq_id__,
__struct_ncs_dom_lim.end_label_seq_id__,
__struct_ref_seq.seq_align_beg__,
__struct_ref_seq.seq_align_end__,
__struct_ref_seq_dif.seq_num__,
__struct_sheet_hbond.range_1_beg_label_seq_id__,
__struct_sheet_hbond.range_1_end_label_seq_id__,
__struct_sheet_hbond.range_2_beg_label_seq_id__,
__struct_sheet_hbond.range_2_end_label_seq_id__,
__struct_sheet_range.beg_label_seq_id__,
__struct_sheet_range.end_label_seq_id__,
__struct_site_gen.label_seq_id__.
The permitted range is $[1, \infty)$. **[entity_poly_seq]**

ENTITY_SRC_GEN

Data items in the ENTITY_SRC_GEN category record details of the source from which the entity was obtained in cases where the source was genetically manipulated. The following are treated separately: items pertaining to the tissue from which the gene was obtained, items pertaining to the host organism for gene expression and items pertaining to the actual producing organism (plasmid).
Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_src_gen.entity_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_src_gen.entity_id
_entity_src_gen.gene_src_common_name
_entity_src_gen.gene_src_genus
_entity_src_gen.gene_src_species
_entity_src_gen.gene_src_strain
_entity_src_gen.host_org_common_name
_entity_src_gen.host_org_genus
_entity_src_gen.host_org_species
_entity_src_gen.plasmid_name
 1  'HIV-1'  ?  ?  'NY-5'
    'bacteria'  'Escherichia'  'coli'  'pB322'
```

*__entity_src_gen.entity_id__
This data item is a pointer to __entity.id__ in the ENTITY category.

__entity_src_gen.gene_src_common_name__ (*text*)
The common name of the natural organism from which the gene was obtained.
Examples: 'man', 'yeast', 'bacteria'. **[entity_src_gen]**

__entity_src_gen.gene_src_details__ (*text*)
A description of special aspects of the natural organism from which the gene was obtained.
[entity_src_gen]

__entity_src_gen.gene_src_genus__ (*text*)
The genus of the natural organism from which the gene was obtained.
Examples: 'Homo', 'Saccharomyces', 'Escherichia'. **[entity_src_gen]**

__entity_src_gen.gene_src_species__ (*text*)
The species of the natural organism from which the gene was obtained.
Examples: 'sapiens', 'cerevisiae', 'coli'. **[entity_src_gen]**

__entity_src_gen.gene_src_strain__ (*text*)
The strain of the natural organism from which the gene was obtained, if relevant.
Examples: 'DH5a', 'BMH 71-18'. **[entity_src_gen]**

__entity_src_gen.gene_src_tissue__ (*text*)
The tissue of the natural organism from which the gene was obtained.
Examples: 'heart', 'liver', 'eye lens'. **[entity_src_gen]**

__entity_src_gen.gene_src_tissue_fraction__ (*text*)
The subcellular fraction of the tissue of the natural organism from which the gene was obtained.
Examples: 'mitochondria', 'nucleus', 'membrane'. **[entity_src_gen]**

__entity_src_gen.host_org_common_name__ (*text*)
The common name of the organism that served as host for the production of the entity.
Examples: 'yeast', 'bacteria'. **[entity_src_gen]**

_entity_src_gen.host_org_details (*text*)
A description of special aspects of the organism that served as host for the production of the entity.

[entity_src_gen]

_entity_src_gen.host_org_genus (*text*)
The genus of the organism that served as host for the production of the entity.
Examples: 'Saccharomyces', 'Escherichia'. **[entity_src_gen]**

_entity_src_gen.host_org_species (*text*)
The species of the organism that served as host for the production of the entity.
Examples: 'cerevisiae', 'coli'. **[entity_src_gen]**

_entity_src_gen.host_org_strain (*text*)
The strain of the organism that served as host for the production of the entity.
Examples: 'DH5a', 'BMH 71-18'. **[entity_src_gen]**

_entity_src_gen.plasmid_details (*text*)
A description of special aspects of the plasmid that produced the entity in the host organism.

[entity_src_gen]

_entity_src_gen.plasmid_name (*text*)
The name of the plasmid that produced the entity in the host organism.
Examples: 'pET3C', 'pT123sab'. **[entity_src_gen]**

ENTITY_SRC_NAT

Data items in the ENTITY_SRC_NAT category record details of the source from which the entity was obtained in cases where the entity was isolated directly from a natural tissue.
Category group(s): **inclusive_group**
entity_group
Category key(s): **_entity_src_nat.entity_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_entity_src_nat.entity_id
_entity_src_nat.common_name
_entity_src_nat.genus
_entity_src_nat.species
_entity_src_nat.details
 2  'bacteria'  'Actinomycetes' ?
;    Acetyl-pepstatin was isolated by Dr. K. Oda, Osaka
     Prefecture University, and provided to us by Dr. Ben
     Dunn, University of Florida, and Dr. J. Kay, University
     of Wales.
;
```

* **_entity_src_nat.common_name** (*text*)
The common name of the organism from which the entity was isolated.
Examples: 'man', 'yeast', 'bacteria'. **[entity_src_nat]**

_entity_src_nat.details (*text*)
A description of special aspects of the organism from which the entity was isolated.

[entity_src_nat]

* **_entity_src_nat.entity_id**
This data item is a pointer to **_entity.id** in the ENTITY category.

* **_entity_src_nat.genus** (*text*)
The genus of the organism from which the entity was isolated.
Examples: 'Homo', 'Saccharomyces', 'Escherichia'. **[entity_src_nat]**

* **_entity_src_nat.species** (*text*)
The species of the organism from which the entity was isolated.
Examples: 'sapiens', 'cerevisiae', 'coli'. **[entity_src_nat]**

* **_entity_src_nat.strain** (*text*)
The strain of the organism from which the entity was isolated.
Examples: 'DH5a', 'BMH 71-18'. **[entity_src_nat]**

* **_entity_src_nat.tissue** (*text*)
The tissue of the organism from which the entity was isolated.
Examples: 'heart', 'liver', 'eye lens'. **[entity_src_nat]**

* **_entity_src_nat.tissue_fraction** (*text*)
The subcellular fraction of the tissue of the organism from which the entity was isolated.
Examples: 'mitochondria', 'nucleus', 'membrane'. **[entity_src_nat]**

ENTRY

There is only one item in the ENTRY category, **_entry.id**. This data item gives a name to this entry and is indirectly a key to the categories (such as CELL, GEOM, EXPTL) that describe information pertinent to the entire data block.
Category group(s): **inclusive_group**
entry_group
Category key(s): **_entry.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_entry.id                                   '5HVP'
```

Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_entry.id                                   'TOZ'
```

* **_entry.id** (*code*)
_audit_block_code (*cif_core.dic 2.0.1*)
The value of **_entry.id** identifies the data block. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_atom_sites.entry_id,
_cell.entry_id,
_cell_measurement.entry_id,
_chemical.entry_id,
_chemical_formula.entry_id,
_computing.entry_id,
_database.entry_id,
_database_PDB_matrix.entry_id,
_entry_link.entry_id,
_exptl.entry_id,
_geom.entry_id,
_journal.entry_id,
_phasing_averaging.entry_id,
_phasing_isomorphous.entry_id,
_phasing_MAD.entry_id,
_phasing_MIR.entry_id,
_publ.entry_id,
_publ_manuscript_incl.entry_id,

_refine.entry_id,
_refine_analyze.entry_id,
_reflns.entry_id,
_struct.entry_id,
_struct_keywords.entry_id,
_struct_mon_details.entry_id,
_symmetry.entry_id. **[entry]**

ENTRY_LINK

Data items in the ENTRY_LINK category record the relationships between the current data block identified by **_entry.id** and other data blocks within the current file which may be referenced in the current data block.

Category group(s): **inclusive_group**
entry_group
Category key(s): **_entry_link.id**
_entry_link.entry_id

Example 1 – example file for the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
_entry_link.id
_entry_link.entry_id
_entry_link.details
 KSE_COM  KSE_TEXT
     'experimental data common to ref./mod. structures'
 KSE_REF  KSE_TEXT  'reference structure'
 KSE_MOD  KSE_TEXT  'modulated structure'
```

_entry_link.details (*text*)

_audit_link_block_description (*cif_core.dic 2.0.1*)

A description of the relationship between the data blocks identified by **_entry_link.id** and **_entry_link.entry_id**.

[entry_link]

*_entry_link.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

*_entry_link.id (*code*)

_audit_link_block_code (*cif_core.dic 2.0.1*)

The value of **_entry_link.id** identifies a data block related to the current data block.

[entry_link]

EXPTL

Data items in the EXPTL category record details about the experimental work prior to the intensity measurements and details about the absorption-correction technique employed.

Category group(s): **inclusive_group**
exptl_group
Category key(s): **_exptl.entry_id**

Example 1 – based on laboratory records for $Yb(S\text{-}C_5H_4N)_2(THF)_4$.

```
_exptl.entry_id                    datablock1
_exptl.absorpt_coefficient_mu      1.22
_exptl.absorpt_correction_T_max    0.896
_exptl.absorpt_correction_T_min    0.802
_exptl.absorpt_correction_type     integration
_exptl.absorpt_process_details
; Gaussian grid method from SHELX76
  Sheldrick, G. M., "SHELX-76: structure determination and
  refinement program", Cambridge University, UK, 1976
;
_exptl.crystals_number             1
_exptl.details
; Enraf-Nonius LT2 liquid nitrogen variable-temperature
  device used
;
_exptl.method            'single-crystal x-ray diffraction'
_exptl.method_details
; graphite monochromatized Cu K(alpha) fixed tube and
  Enraf-Nonius CAD4 diffractometer used
;
```

_exptl.absorpt_coefficient_mu (*float*)

_exptl_absorpt_coefficient_mu (*cif_core.dic 2.0.1*)

The absorption coefficient μ in reciprocal millimetres calculated from the atomic content of the cell, the density and the radiation wavelength.

The permitted range is $[0.0, \infty)$. **[exptl]**

_exptl.absorpt_correction_T_max (*float*)

_exptl_absorpt_correction_T_max (*cif_core.dic 2.0.1*)

The maximum transmission factor for the crystal and radiation. The maximum and minimum transmission factors are also referred to as the absorption correction A or $1/A^*$.

The permitted range is [0.0, 1.0]. **[exptl]**

_exptl.absorpt_correction_T_min (*float*)

_exptl_absorpt_correction_T_min (*cif_core.dic 2.0.1*)

The minimum transmission factor for the crystal and radiation. The maximum and minimum transmission factors are also referred to as the absorption correction A or $1/A^*$.

The permitted range is [0.0, 1.0]. **[exptl]**

_exptl.absorpt_correction_type (*ucode*)

_exptl_absorpt_correction_type (*cif_core.dic 2.0.1*)

The absorption correction type and method. The value 'empirical' should *not* be used unless more detailed information is not available.

The data value must be one of the following:

analytical	analytical from crystal shape
cylinder	cylindrical
empirical	empirical from intensities
gaussian	Gaussian from crystal shape
integration	integration from crystal shape
multi-scan	symmetry-related measurements
none	no correction applied
numerical	numerical from crystal shape
psi-scan	ψ-scan corrections
refdelf	refined from ΔF
sphere	spherical

[exptl]

_exptl.absorpt_process_details (*text*)

_exptl_absorpt_process_details (*cif_core.dic 2.0.1*)

Description of the absorption process applied to the intensities. A literature reference should be supplied for ψ-scan techniques.

Example: 'Tompa analytical'. **[exptl]**

_exptl.crystals_number (*int*)

_exptl_crystals_number (*cif_core.dic 2.0.1*)

The total number of crystals used in the measurement of intensities.

The permitted range is $[1, \infty)$. **[exptl]**

_exptl.details (*text*)

_exptl_special_details (*cif_core.dic 2.0.1*)

Any special information about the experimental work prior to the intensity measurement. See also **_exptl_crystal.preparation**.

[exptl]

*_exptl.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

*_exptl.method (*line*)

The method used in the experiment.

Examples: 'single-crystal x-ray diffraction',
'single-crystal neutron diffraction',
'single-crystal electron diffraction', 'fiber x-ray diffraction',
'fiber neutron diffraction', 'fiber electron diffraction',
'single-crystal joint x-ray and neutron diffraction',
'single-crystal joint x-ray and electron diffraction',
'solution nmr', 'solid-state nmr', 'theoretical model', 'other'.

[exptl]

_exptl.method_details (*text*)

A description of special aspects of the experimental method.

Examples: '29 structures', 'minimized average structure'. **[exptl]**

EXPTL_CRYSTAL

Data items in the EXPTL_CRYSTAL category record the results of experimental measurements on the crystal or crystals used, such as shape, size or density.

Category group(s): **inclusive_group**
exptl_group

Category key(s): **_exptl_crystal.id**

Example 1 – based on laboratory records for $Yb(S\text{-}C_5H_4N)_2(THF)_4$.

```
_exptl_crystal.id                  xst21
_exptl_crystal.colour              'pale yellow'
_exptl_crystal.density_diffrn      1.113
_exptl_crystal.density_Matthews    1.01
_exptl_crystal.density_meas        1.11
_exptl_crystal.density_meas_temp   294.5
_exptl_crystal.density_method      'neutral buoyancy'
_exptl_crystal.density_percent_sol 0.15
                                # P = 1 - (1.23*N*MMass) / V
_exptl_crystal.description         'hexagonal rod, uncut'
_exptl_crystal.F_000               202
_exptl_crystal.preparation
; hanging drop, crystal soaked in 10% ethylene glycol for
  10 h, then placed in nylon loop at data collection time
;
_exptl_crystal.size_max            0.30
_exptl_crystal.size_mid            0.20
_exptl_crystal.size_min            0.05
_exptl_crystal.size_rad            0.025
```

Example 2 - using separate items to define upper and lower limits for a value.

```
_exptl_crystal.density_meas_gt      2.5
_exptl_crystal.density_meas_lt      5.0
```

Example 3 - here the density was measured at some unspecified temperature below room temperature.

```
_exptl_crystal.density_meas_temp_lt  300
```

_exptl_crystal.colour (*line*)

_exptl_crystal_colour (*cif_core.dic 2.0.1*)

The colour of the crystal.

Example: 'dark green'. **[exptl_crystal]**

_exptl_crystal.colour_lustre (*line*)

_exptl_crystal_colour_lustre (*cif_core.dic 2.3*)

The enumeration list of standardized names developed for the International Centre for Diffraction Data. The colour of a crystal is given by the combination of **_exptl_crystal.colour_modifier** with **_exptl_crystal.colour_primary**, as in 'dark-green' or 'bluish-violet', if necessary combined with **_exptl_crystal.colour_lustre**, as in 'metallic-green'.

Related item: **_exptl_crystal.colour** (alternate).

The data value must be one of the following:

metallic
dull
clear **[exptl_crystal]**

_exptl_crystal.colour_modifier (*line*)

_exptl_crystal_colour_modifier (*cif_core.dic 2.3*)

The enumeration list of standardized names developed for the International Centre for Diffraction Data. The colour of a crystal is given by the combination of **_exptl_crystal.colour_modifier** with **_exptl_crystal.colour_primary**, as in 'dark-green' or 'bluish-violet', if necessary combined with **_exptl_crystal.colour_lustre**, as in 'metallic-green'.

Related item: **_exptl_crystal.colour** (alternate).

The data value must be one of the following:

light
dark
whitish
blackish
grayish
brownish
reddish
pinkish
orangish
yellowish
greenish
bluish **[exptl_crystal]**

_exptl_crystal.colour_primary (*line*)

_exptl_crystal_colour_primary (*cif_core.dic 2.3*)

The enumeration list of standardized names developed for the International Centre for Diffraction Data. The colour of a crystal is given by the combination of **_exptl_crystal.colour_modifier** with **_exptl_crystal.colour_primary**, as in 'dark-green' or 'bluish-violet', if necessary combined with **_exptl_crystal.colour_lustre**, as in 'metallic-green'.

Related item: **_exptl_crystal.colour** (alternate).

The data value must be one of the following:

colourless
white
black
gray
brown
red
pink
orange
yellow
green
blue
violet **[exptl_crystal]**

_exptl_crystal.density_diffrn (*float*)

_exptl_crystal_density_diffrn (*cif_core.dic 2.0.1*)

Density values calculated from the crystal cell and contents. The units are megagrams per cubic metre (grams per cubic centimetre).

The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

_exptl_crystal.density_Matthews (*float*)

The density of the crystal, expressed as the ratio of the volume of the asymmetric unit to the molecular mass of a monomer of the structure, in units of ångströms^3 per dalton.

Reference: Matthews, B. W. (1968). *J. Mol. Biol.* **33**, 491–497.

[exptl_crystal]

_exptl_crystal.density_meas (*float, su*)

_exptl_crystal_density_meas (*cif_core.dic 2.3*)

Density values measured using standard chemical and physical methods. The units are megagrams per cubic metre (grams per cubic centimetre).

The permitted range is $[0.0, \infty)$.

Related item: **_exptl_crystal.density_meas_esd** (associated esd).

[exptl_crystal]

_exptl_crystal.density_meas_esd (*float*)

The standard uncertainty (estimated standard deviation) of **_exptl_crystal.density_meas**.

Related item: **_exptl_crystal.density_meas** (associated value).

[exptl_crystal]

_exptl_crystal.density_meas_gt (*float*)
_exptl_crystal_density_meas_gt (*cif_core.dic 2.3*)
The value above which the density measured using standard chemical and physical methods lies. The units are megagrams per cubic metre (grams per cubic centimetre). **_exptl_crystal.density_meas_gt** and **_exptl_crystal.density_meas_lt** should not be used to report new experimental work, for which **_exptl_crystal.density_meas** should be used. These items are intended for use in reporting information in existing databases and archives which would be misleading if reported under **_exptl_crystal.density_meas**.
The permitted range is $[0.0, \infty)$.
Related item: **_exptl_crystal.density_meas** (alternate) .
Example: '2.5' (lower limit for the density (only the range within which the density lies was given in the original paper)). **[exptl_crystal]**

_exptl_crystal.density_meas_lt (*float*)
_exptl_crystal_density_meas_lt (*cif_core.dic 2.3*)
The value below which the density measured using standard chemical and physical methods lies. The units are megagrams per cubic metre (grams per cubic centimetre). **_exptl_crystal.density_meas_gt** and **_exptl_crystal.density_meas_lt** should not be used to report new experimental work, for which **_exptl_crystal.density_meas** should be used. These items are intended for use in reporting information in existing databases and archives which would be misleading if reported under **_exptl_crystal.density_meas**.
The permitted range is $[0.0, \infty)$.
Related item: **_exptl_crystal.density_meas** (alternate) .
Examples: '1.0' (specimen floats in water), '5.0' (upper limit for the density (only the range within which the density lies was given in the original paper)). **[exptl_crystal]**

_exptl_crystal.density_meas_temp (*float, su*)
_exptl_crystal_density_meas_temp (*cif_core.dic 2.3*)
Temperature in kelvins at which **_exptl_crystal.density_meas** was determined.
The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

_exptl_crystal.density_meas_temp_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_exptl_crystal.density_meas_temp**.
[exptl_crystal]

_exptl_crystal.density_meas_temp_gt (*float*)
_exptl_crystal_density_meas_temp_gt (*cif_core.dic 2.3*)
Temperature in kelvins above which **_exptl_crystal.density_meas** was determined. **_exptl_crystal.density_meas_temp_gt** and **_exptl_crystal.density_meas_temp_lt** should not be used for reporting new work, for which the correct temperature of measurement should be given. These items are intended for use in reporting information stored in databases or archives which would be misleading if reported under **_exptl_crystal.density_meas_temp**.
The permitted range is $[0.0, \infty)$.
Related item: **_exptl_crystal.density_meas_temp** (alternate).
[exptl_crystal]

_exptl_crystal.density_meas_temp_lt (*float*)
_exptl_crystal_density_meas_temp_lt (*cif_core.dic 2.3*)
Temperature in kelvins below which **_exptl_crystal.density_meas** was determined. **_exptl_crystal.density_meas_temp_gt** and **_exptl_crystal.density_meas_temp_lt** should not be used for reporting new work, for which the correct temperature of measurement should be given. These items are intended for use in reporting information stored in databases or archives which would be misleading if reported under **_exptl_crystal.density_meas_temp**.
The permitted range is $[0.0, \infty)$.
Related item: **_exptl_crystal.density_meas_temp** (alternate) .
Example: '300' (The density was measured at some unspecified temperature below room temperature.). **[exptl_crystal]**

_exptl_crystal.density_method (*text*)
_exptl_crystal_density_method (*cif_core.dic 2.0.1*)
The method used to measure **_exptl_crystal.density_meas**.
[exptl_crystal]

_exptl_crystal.density_percent_sol (*float*)
Density value P calculated from the crystal cell and contents, expressed as per cent solvent.

$$P = 1 - (1.23 N M_{\mathrm{Mass}})/V,$$

where N = the number of molecules in the unit cell, M_{Mass} = the molecular mass of each molecule (g mol^{-1}), V = the volume of the unit cell (Å^3) and 1.23 = a conversion factor evaluated as

$$\frac{(0.74 \mathrm{cm}^3/\mathrm{g})(10^{24} \mathrm{Å}^3/\mathrm{cm}^3)}{(6.02 \times 10^{23} \mathrm{molecules/mole})},$$

where 0.74 is an assumed value for the partial specific volume of the molecule.
The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

_exptl_crystal.description (*text*)
_exptl_crystal_description (*cif_core.dic 2.0.1*)
A description of the quality and habit of the crystal. The crystal dimensions should not normally be reported here; use instead the specific items in the EXPTL_CRYSTAL category relating to size for the gross dimensions of the crystal and data items in the EXPTL_CRYSTAL_FACE category to describe the relationship between individual faces.
[exptl_crystal]

_exptl_crystal.F_000 (*int*)
_exptl_crystal_F_000 (*cif_core.dic 2.0.1*)
The effective number of electrons in the crystal unit cell contributing to $F(000)$. This may contain dispersion contributions and is calculated as

$$F(000) = \left[\left(\sum f_r \right)^2 + \left(\sum f_i \right)^2 \right]^{1/2},$$

where f_r = real part of the scattering factors at $\theta = 0°$, f_i = imaginary part of the scattering factors at $\theta = 0°$ and the sum is taken over each atom in the unit cell.
The permitted range is $[1, \infty)$. **[exptl_crystal]**

***_exptl_crystal.id** (*code*)
_exptl_crystal_id (*cif_core.dic 2.0.1*)
The value of **_exptl_crystal.id** must uniquely identify a record in the EXPTL_CRYSTAL list. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_diffrn.crystal_id,
_exptl_crystal_grow.crystal_id,
_exptl_crystal_face.crystal_id,
_exptl_crystal_grow_comp.crystal_id,
_refln.crystal_id. **[exptl_crystal]**

_exptl_crystal.preparation (*text*)
_exptl_crystal_preparation (*cif_core.dic 2.0.1*)
Details of crystal growth and preparation of the crystal (*e.g.* mounting) prior to the intensity measurements.
Example: 'mounted in an argon-filled quartz capillary'.
[exptl_crystal]

_exptl_crystal.size_max (*float*)
_exptl_crystal_size_max (*cif_core.dic 2.0.1*)
The maximum dimension of the crystal. This item may appear in a list with **_exptl_crystal.id** if multiple crystals are used in the experiment.
The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

_exptl_crystal.size_mid (*float*)
_exptl_crystal_size_mid (*cif_core.dic 2.0.1*)
The medial dimension of the crystal. This item may appear in a list with **_exptl_crystal.id** if multiple crystals are used in the experiment.
The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

_exptl_crystal.size_min (*float*)
_exptl_crystal_size_min (*cif_core.dic 2.0.1*)
The minimum dimension of the crystal. This item may appear in a list with **_exptl_crystal.id** if multiple crystals are used in the experiment.
The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

_exptl_crystal.size_rad (*float*)
_exptl_crystal_size_rad (*cif_core.dic 2.0.1*)
The radius of the crystal, if the crystal is a sphere or a cylinder. This item may appear in a list with **_exptl_crystal.id** if multiple crystals are used in the experiment.
The permitted range is $[0.0, \infty)$. **[exptl_crystal]**

EXPTL_CRYSTAL_FACE

Data items in the EXPTL_CRYSTAL_FACE category record details of the crystal faces.
Category group(s): **inclusive_group**
exptl_group
Category key(s): **_exptl_crystal_face.crystal_id**
_exptl_crystal_face.index_h
_exptl_crystal_face.index_k
_exptl_crystal_face.index_l

Example 1 – based on laboratory records for $Yb(S\text{-}C_5H_4N)_2(THF)_4$ for the 100 face of crystal xstl1.

```
_exptl_crystal_face.crystal_id    xstl1
_exptl_crystal_face.index_h       1
_exptl_crystal_face.index_k       0
_exptl_crystal_face.index_l       0
_exptl_crystal_face.diffr_chi     42.56
_exptl_crystal_face.diffr_kappa   30.23
_exptl_crystal_face.diffr_phi     -125.56
_exptl_crystal_face.diffr_psi     -0.34
_exptl_crystal_face.perp_dist     0.025
```

* **_exptl_crystal_face.crystal_id**
This data item is a pointer to **_exptl_crystal.id** in the EXPTL_CRYSTAL category.

_exptl_crystal_face.diffr_chi (*float*)
_exptl_crystal_face_diffr_chi (*cif_core.dic 2.0.1*)
The χ diffractometer setting angle in degrees for a specific crystal face associated with **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

_exptl_crystal_face.diffr_kappa (*float*)
_exptl_crystal_face_diffr_kappa (*cif_core.dic 2.0.1*)
The κ diffractometer setting angle in degrees for a specific crystal face associated with **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

_exptl_crystal_face.diffr_phi (*float*)
_exptl_crystal_face_diffr_phi (*cif_core.dic 2.0.1*)
The φ diffractometer setting angle in degrees for a specific crystal face associated with **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

_exptl_crystal_face.diffr_psi (*float*)
_exptl_crystal_face_diffr_psi (*cif_core.dic 2.0.1*)
The ψ diffractometer setting angle in degrees for a specific crystal face associated with **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

* **_exptl_crystal_face.index_h** (*int*)
_exptl_crystal_face_index_h (*cif_core.dic 2.0.1*)
Miller index h of the crystal face associated with the value **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

* **_exptl_crystal_face.index_k** (*int*)
_exptl_crystal_face_index_k (*cif_core.dic 2.0.1*)
Miller index k of the crystal face associated with the value **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

* **_exptl_crystal_face.index_l** (*int*)
_exptl_crystal_face_index_l (*cif_core.dic 2.0.1*)
Miller index l of the crystal face associated with the value **_exptl_crystal_face.perp_dist**.
[exptl_crystal_face]

_exptl_crystal_face.perp_dist (*float*)
_exptl_crystal_face_perp_dist (*cif_core.dic 2.0.1*)
The perpendicular distance in millimetres from the face to the centre of rotation of the crystal.
The permitted range is $[0.0, \infty)$. **[exptl_crystal_face]**

EXPTL_CRYSTAL_GROW

Data items in the EXPTL_CRYSTAL_GROW category record details about the conditions and methods used to grow the crystal.
Category group(s): **inclusive_group**
exptl_group
Category key(s): **_exptl_crystal_grow.crystal_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_exptl_crystal_grow.crystal_id     1
_exptl_crystal_grow.method         'hanging drop'
_exptl_crystal_grow.apparatus      'Linbro plates'
_exptl_crystal_grow.atmosphere     'room air'
_exptl_crystal_grow.pH             4.7
_exptl_crystal_grow.temp           18(3)
_exptl_crystal_grow.time           'approximately 2 days'
```

_exptl_crystal_grow.apparatus (*text*)
The physical apparatus in which the crystal was grown.
Examples: 'Linbro plate', 'sandwich box', 'ACA plates'.
[exptl_crystal_grow]

_exptl_crystal_grow.atmosphere (*text*)
The nature of the gas or gas mixture in which the crystal was grown.
Examples: 'room air', 'nitrogen', 'argon'. **[exptl_crystal_grow]**

*_exptl_crystal_grow.crystal_id
This data item is a pointer to **_exptl_crystal.id** in the EXPTL_CRYSTAL category.

_exptl_crystal_grow.details (*text*)
A description of special aspects of the crystal growth.
Examples:
```
; Solution 2 was prepared as a well solution and
  mixed. A droplet containing 2 \ml of solution
  1 was delivered onto a cover slip; 2 \ml of
  solution 2 was added to the droplet without
  mixing.
;
; Crystal plates were originally stored at room
  temperature for 1 week but no nucleation
  occurred. They were then transferred to 4
  degrees C, at which temperature well formed
  single crystals grew in 2 days.
;
; The dependence on pH for successful crystal
  growth is very sharp. At pH 7.4 only showers
  of tiny crystals grew, at pH 7.5 well formed
  single crystals grew, at pH 7.6 no
  crystallization occurred at all.
;
```
[exptl_crystal_grow]

_exptl_crystal_grow.method (*text*)
The method used to grow the crystals.
Examples: 'batch precipitation', 'batch dialysis', 'hanging drop vapor diffusion', 'sitting drop vapor diffusion'.
[exptl_crystal_grow]

_exptl_crystal_grow.method_ref (*text*)
A literature reference that describes the method used to grow the crystals.
Example: 'McPherson et al., 1988'. **[exptl_crystal_grow]**

_exptl_crystal_grow.pH (*float*)
The pH at which the crystal was grown. If more than one pH was employed during the crystallization process, the final pH should be noted here and the protocol involving multiple pH values should be described in **_exptl_crystal_grow.details**.
The permitted range is $[0.0, \infty)$.
Examples: '7.4', '7.6', '4.3'. **[exptl_crystal_grow]**

_exptl_crystal_grow.pressure (*float, su*)
The ambient pressure in kilopascals at which the crystal was grown.
The permitted range is $[0.0, \infty)$.
Related item: **_exptl_crystal_grow.pressure_esd** (associated esd).
[exptl_crystal_grow]

_exptl_crystal_grow.pressure_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_exptl_crystal_grow.pressure**.
Related item: **_exptl_crystal_grow.pressure** (associated value).
[exptl_crystal_grow]

_exptl_crystal_grow.seeding (*text*)
A description of the protocol used for seeding the crystal growth.
Examples: 'macroseeding',
```
; Microcrystals were introduced from a previous
  crystal growth experiment by transfer with a
  human hair.
;
```
[exptl_crystal_grow]

_exptl_crystal_grow.seeding_ref (*text*)
A literature reference that describes the protocol used to seed the crystal.
Example: 'Stura et al., 1989'. **[exptl_crystal_grow]**

_exptl_crystal_grow.temp (*float, su*)
The temperature in kelvins at which the crystal was grown. If more than one temperature was employed during the crystallization process, the final temperature should be noted here and the protocol involving multiple temperatures should be described in **_exptl_crystal_grow.details**.
The permitted range is $[0.0, \infty)$.
Related item: **_exptl_crystal_grow.temp_esd** (associated esd).
[exptl_crystal_grow]

_exptl_crystal_grow.temp_details (*text*)
A description of special aspects of temperature control during crystal growth.
[exptl_crystal_grow]

_exptl_crystal_grow.temp_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_exptl_crystal_grow.temp**.
Related item: **_exptl_crystal_grow.temp** (associated value).
[exptl_crystal_grow]

_exptl_crystal_grow.time (*text*)
The approximate time that the crystal took to grow to the size used for data collection.
Examples: 'overnight', '2-4 days', '6 months'. **[exptl_crystal_grow]**

EXPTL_CRYSTAL_GROW_COMP

Data items in the EXPTL_CRYSTAL_GROW_COMP category record details about the components of the solutions that were 'mixed' (by whatever means) to produce the crystal. In general, solution 1 is the solution that contains the molecule to be crystallized and solution 2 is the solution that contains the precipitant. However, the number of solutions required to describe the crystallization protocol is not limited to 2. Details of the crystallization protocol should be given in **_exptl_crystal_grow_comp.details** using the solutions described in EXPTL_CRYSTAL_GROW_COMP.
Category group(s): **inclusive_group**
exptl_group
Category key(s): **_exptl_crystal_grow_comp.id**
_exptl_crystal_grow_comp.crystal_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_exptl_crystal_grow_comp.crystal_id
_exptl_crystal_grow_comp.id
_exptl_crystal_grow_comp.sol_id
_exptl_crystal_grow_comp.name
_exptl_crystal_grow_comp.volume
_exptl_crystal_grow_comp.conc
_exptl_crystal_grow_comp.details
 1 1  1  'HIV-1 protease'  '0.002 ml'  '6 mg/ml'
; The protein solution was in a buffer containing 25 mM NaCl,
  100 mM NaMES/ MES buffer, pH 7.5, 3 mM NaAzide
;
 1 2 2 'NaCl'          '0.200 ml'  '4    M' 'in 3 mM NaAzide'
 1 3 2 'Acetic Acid'   '0.047 ml'  '100 mM' 'in 3 mM NaAzide'
 1 4 2 'Na Acetate'    '0.053 ml'  '100 mM'
; in 3 mM NaAzide. Buffer components were mixed to produce a
  pH of 4.7 according to a ratio calculated from the pKa. The
  actual pH of solution 2 was not measured.
;
 1 5 2 'water'         '0.700 ml'   'neat'  'in 3 mM NaAzide'
```

_exptl_crystal_grow_comp.conc (*line*)
The concentration of the solution component.
Examples: '200 \ml', '0.1 ml'. **[exptl_crystal_grow_comp]**

*_exptl_crystal_grow_comp.crystal_id
This data item is a pointer to **_exptl_crystal.id** in the EXPTL_CRYSTAL category.

_exptl_crystal_grow_comp.details (*text*)
A description of any special aspects of the solution component. When the solution component is the one that contains the macromolecule, this could be the specification of the buffer in which the macromolecule was stored. When the solution component is a buffer component, this could be the methods (or formula) used to achieve a desired pH.
Examples: 'in 3 mM NaAzide',

```
; The protein solution was in a buffer
  containing 25 mM NaCl, 100 mM NaMES/MES
  buffer, pH 7.5, 3 mM NaAzide
;
; in 3 mM NaAzide. Buffer components were mixed
  to produce a pH of 4.7 according to a ratio
  calculated from the pKa. The actual pH of
  solution 2 was not measured.
;
```
[exptl_crystal_grow_comp]

*_exptl_crystal_grow_comp.id (*line*)
The value of **_exptl_crystal_grow_comp.id** must uniquely identify each item in the EXPTL_CRYSTAL_GROW_COMP list. Note that this item need not be a number; it can be any unique identifier.
Examples: '1', 'A', 'protein in buffer'. **[exptl_crystal_grow_comp]**

_exptl_crystal_grow_comp.name (*line*)
A common name for the component of the solution.
Examples: 'protein in buffer', 'acetic acid'.
[exptl_crystal_grow_comp]

_exptl_crystal_grow_comp.sol_id (*line*)
An identifier for the solution to which the given solution component belongs.
Examples: '1', 'well solution', 'solution A'. **[exptl_crystal_grow_comp]**

_exptl_crystal_grow_comp.volume (*line*)
The volume of the solution component.
Examples: '200 \ml', '0.1 ml'. **[exptl_crystal_grow_comp]**

GEOM

Data items in the GEOM and related (GEOM_ANGLE, GEOM_BOND, GEOM_CONTACT, GEOM_HBOND and GEOM_TORSION) categories record details about the molecular geometry as calculated from the contents of the ATOM, CELL and SYMMETRY data. Geometry data are therefore redundant, in that they can be calculated from other more fundamental quantities in the data block. However, they provide a check on the correctness of both sets of data and enable the most important geometric data to be identified for publication by setting the appropriate publication flag.
Category group(s): **inclusive_group**
geom_group
Category key(s): **_geom.entry_id**

_geom.details (*text*)
_geom_special_details (*cif_core.dic 2.0.1*)
A description of geometry not covered by the existing data names in the GEOM categories, such as least-squares planes.
[geom]

*_geom.entry_id
This data item is a pointer to **_entry.id** in the ENTRY category.

GEOM_ANGLE

Data items in the GEOM_ANGLE category record details about the bond angles as calculated from the contents of the ATOM, CELL and SYMMETRY data.
Category group(s): **inclusive_group**
geom_group
Category key(s): **_geom_angle.atom_site_id_1**
_geom_angle.atom_site_id_2
_geom_angle.atom_site_id_3
_geom_angle.site_symmetry_1
_geom_angle.site_symmetry_2
_geom_angle.site_symmetry_3

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
loop_
_geom_angle.atom_site_id_1
_geom_angle.atom_site_id_2
_geom_angle.atom_site_id_3
_geom_angle.value
_geom_angle.value_esd
_geom_angle.site_symmetry_1
_geom_angle.site_symmetry_2
_geom_angle.site_symmetry_3
_geom_angle.publ_flag
 C2  O1  C5   111.6  0.2  1_555  1_555  1_555  yes
 O1  C2  C3   110.9  0.2  1_555  1_555  1_555  yes
 O1  C2  O21  122.2  0.3  1_555  1_555  1_555  yes
 C3  C2  O21  127.0  0.3  1_555  1_555  1_555  yes
 C2  C3  N4   101.3  0.2  1_555  1_555  1_555  yes
 C2  C3  C31  111.3  0.2  1_555  1_555  1_555  yes
 C2  C3  H3   107    1    1_555  1_555  1_555  no
 N4  C3  C31  116.7  0.2  1_555  1_555  1_555  yes
# - - - - data truncated for brevity - - - -
```

_geom_angle.atom_site_auth_asym_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_asym_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_asym_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_atom_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_atom_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_atom_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_comp_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_comp_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_comp_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_seq_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_seq_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_angle.atom_site_auth_seq_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_geom_angle.atom_site_id_1**
_geom_angle_atom_site_label_1 (*cif_core.dic 2.0.1*)
The identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

* **_geom_angle.atom_site_id_2**
_geom_angle_atom_site_label_2 (*cif_core.dic 2.0.1*)
The identifier of the second of the three atom sites that define the angle. The second atom is taken to be the apex of the angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

* **_geom_angle.atom_site_id_3**
_geom_angle_atom_site_label_3 (*cif_core.dic 2.0.1*)
The identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

_geom_angle.atom_site_label_alt_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_alt_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_alt_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_asym_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_asym_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_asym_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_atom_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_atom_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_atom_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_comp_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_comp_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_comp_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_seq_id_1
An optional identifier of the first of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_seq_id_2
An optional identifier of the second of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_angle.atom_site_label_seq_id_3
An optional identifier of the third of the three atom sites that define the angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_angle.publ_flag (*ucode*)
_geom_angle_publ_flag (*cif_core.dic 2.0.1*)
This code signals whether the angle is referred to in a publication or should be placed in a table of significant angles.
The data value must be one of the following:

no	do not include angle in special list
n	abbreviation for 'no'
yes	do include angle in special list
y	abbreviation for 'yes'

[geom_angle]

***_geom_angle.site_symmetry_1** (*symop*)
_geom_angle_site_symmetry_1 (*cif_core.dic 2.0.1*)
The symmetry code of the first of the three atom sites that define the angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_angle]**

***_geom_angle.site_symmetry_2** (*symop*)
_geom_angle_site_symmetry_2 (*cif_core.dic 2.0.1*)
The symmetry code of the second of the three atom sites that define the angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_angle]**

***_geom_angle.site_symmetry_3** (*symop*)
_geom_angle_site_symmetry_3 (*cif_core.dic 2.0.1*)
The symmetry code of the third of the three atom sites that define the angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_angle]**

_geom_angle.value (*float, su*)
_geom_angle (*cif_core.dic 2.0.1*)
Angle in degrees defined by the three sites **_geom_angle.atom_site_id_1**, **_geom_angle.atom_site_id_2** and **_geom_angle.atom_site_id_3**.
Related item: **_geom_angle.value_esd** (associated esd). **[geom_angle]**

_geom_angle.value_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_geom_angle.value**.
Related item: **_geom_angle.value** (associated value). **[geom_angle]**

GEOM_BOND

Data items in the GEOM_BOND category record details about the bond lengths as calculated from the contents of the ATOM, CELL and SYMMETRY data.
Category group(s): **inclusive_group**
geom_group
Category key(s): **_geom_bond.atom_site_id_1**
_geom_bond.atom_site_id_2
_geom_bond.site_symmetry_1
_geom_bond.site_symmetry_2

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
loop_
_geom_bond.atom_site_id_1
_geom_bond.atom_site_id_2
_geom_bond.dist
_geom_bond.dist_esd
_geom_bond.site_symmetry_1
_geom_bond.site_symmetry_2
_geom_bond.publ_flag
  O1  C2   1.342 0.004  1_555  1_555  yes
  O1  C5   1.439 0.003  1_555  1_555  yes
  C2  C3   1.512 0.004  1_555  1_555  yes
  C2  O21  1.199 0.004  1_555  1_555  yes
  C3  N4   1.465 0.003  1_555  1_555  yes
  C3  C31  1.537 0.004  1_555  1_555  yes
  C3  H3   1.00  0.03   1_555  1_555  no
  N4  C5   1.472 0.003  1_555  1_555  yes
# - - - - data truncated for brevity - - - -
```

_geom_bond.atom_site_auth_asym_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_asym_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_atom_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_atom_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_comp_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_comp_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_seq_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_bond.atom_site_auth_seq_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

***_geom_bond.atom_site_id_1**
_geom_bond_atom_site_label_1 (*cif_core.dic 2.0.1*)
The identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

***_geom_bond.atom_site_id_2**
_geom_bond_atom_site_label_2 (*cif_core.dic 2.0.1*)
The identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

_geom_bond.atom_site_label_alt_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_alt_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_asym_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_asym_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_atom_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_atom_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_comp_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_comp_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_seq_id_1
An optional identifier of the first of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_bond.atom_site_label_seq_id_2
An optional identifier of the second of the two atom sites that define the bond. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_bond.dist (*float, su*)
_geom_bond_distance (*cif_core.dic 2.0.1*)
The intramolecular bond distance in ångströms.
The permitted range is $[0.0, \infty)$.
Related item: **_geom_bond.dist_esd** (associated esd). **[geom_bond]**

_geom_bond.dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_geom_bond.dist**.
Related item: **_geom_bond.dist** (associated value). **[geom_bond]**

_geom_bond.publ_flag (*ucode*)
_geom_bond_publ_flag (*cif_core.dic 2.0.1*)
This code signals whether the bond distance is referred to in a publication or should be placed in a list of significant bond distances.
The data value must be one of the following:

`no`	do not include bond in special list
`n`	abbreviation for 'no'
`yes`	do include bond in special list
`y`	abbreviation for 'yes'

[geom_bond]

***_geom_bond.site_symmetry_1** (*symop*)
_geom_bond_site_symmetry_1 (*cif_core.dic 2.0.1*)
The symmetry code of the first of the two atom sites that define the bond.
Where no value is given, the assumed value is '`1_555`'.
Examples: '`.`' (no symmetry or translation to site), '`4`' (4th symmetry operation applied), '`7_645`' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_bond]**

***_geom_bond.site_symmetry_2** (*symop*)
_geom_bond_site_symmetry_2 (*cif_core.dic 2.0.1*)
The symmetry code of the second of the two atom sites that define the bond.
Where no value is given, the assumed value is '`1_555`'.
Examples: '`.`' (no symmetry or translation to site), '`4`' (4th symmetry operation applied), '`7_645`' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_bond]**

_geom_bond.valence (*int*)
_geom_bond_valence (*cif_core.dic 2.3*)
The bond valence calculated from **_geom_bond.dist**.
[geom_bond]

GEOM_CONTACT

Data items in the GEOM_CONTACT category record details about interatomic contacts as calculated from the contents of the ATOM, CELL and SYMMETRY data.

Category group(s): **inclusive_group**
geom_group
Category key(s): **_geom_contact.atom_site_id_1**
_geom_contact.atom_site_id_2
_geom_contact.site_symmetry_1
_geom_contact.site_symmetry_2

*Example 1 – based on data set CLPHO6 of Ferguson, Ruhl, McKervey & Browne [Acta Cryst. (1992), C**48**, 2262–2264].*

```
loop_
_geom_contact.atom_site_id_1
_geom_contact.atom_site_id_2
_geom_contact.dist
_geom_contact.dist_esd
_geom_contact.site_symmetry_1
_geom_contact.site_symmetry_2
_geom_contact.publ_flag
 O(1)  O(2)   2.735 0.003 .  .  yes
 H(O1) O(2)   1.82  .     .  .  no
```

_geom_contact.atom_site_auth_asym_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_asym_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_atom_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_atom_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_comp_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_comp_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_seq_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_contact.atom_site_auth_seq_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_geom_contact.atom_site_id_1**
_geom_contact_atom_site_label_1 (*cif_core.dic 2.0.1*)
The identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

* **_geom_contact.atom_site_id_2**
_geom_contact_atom_site_label_2 (*cif_core.dic 2.0.1*)
The identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

_geom_contact.atom_site_label_alt_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_alt_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_asym_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_asym_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_atom_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_atom_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_comp_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_comp_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_seq_id_1
An optional identifier of the first of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_contact.atom_site_label_seq_id_2
An optional identifier of the second of the two atom sites that define the contact. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_contact.dist (*float, su*)
_geom_contact_distance (*cif_core.dic 2.0.1*)
The interatomic contact distance in ångströms.
The permitted range is $[0.0, \infty)$.
Related item: **_geom_contact.dist_esd** (associated esd). **[geom_contact]**

_geom_contact.dist_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_geom_contact.dist**.
Related item: **_geom_contact.dist** (associated value). **[geom_contact]**

_geom_contact.publ_flag (*ucode*)
_geom_contact_publ_flag (*cif_core.dic 2.0.1*)
This code signals whether the contact distance is referred to in a publication or should be placed in a list of significant contact distances.
The data value must be one of the following:

no	do not include distance in special list
n	abbreviation for 'no'
yes	do include distance in special list
y	abbreviation for 'yes'

[geom_contact]

*__geom_contact.site_symmetry_1** (*symop*)
_geom_contact_site_symmetry_1 (*cif_core.dic 2.0.1*)
The symmetry code of the first of the two atom sites that define the contact.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_contact]**

*__geom_contact.site_symmetry_2** (*symop*)
_geom_contact_site_symmetry_2 (*cif_core.dic 2.0.1*)
The symmetry code of the second of the two atom sites that define the contact.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_contact]**

GEOM_HBOND

Data items in the GEOM_HBOND category record details about hydrogen bonds as calculated from the contents of the ATOM, CELL and SYMMETRY data.
Category group(s): **inclusive_group** **geom_group**
Category key(s): **_geom_hbond.atom_site_id_A** **_geom_hbond.atom_site_id_D** **_geom_hbond.atom_site_id_H** **_geom_hbond.site_symmetry_A** **_geom_hbond.site_symmetry_D** **_geom_hbond.site_symmetry_H**

*Example 1 – based on $C_{14}H_{13}ClN_2O.H_2O$, reported by Palmer, Puddle & Lisgarten [Acta Cryst. (1993), C**49**, 1777–1779].*

```
loop_
_geom_hbond.atom_site_id_D
_geom_hbond.atom_site_id_H
_geom_hbond.atom_site_id_A
_geom_hbond.dist_DH
_geom_hbond.dist_HA
_geom_hbond.dist_DA
_geom_hbond.angle_DHA
_geom_hbond.publ_flag
N6   HN6  OW   0.888  1.921  2.801  169.6  yes
OW   HO2  O7   0.917  1.923  2.793  153.5  yes
OW   HO1  N10  0.894  1.886  2.842  179.7  yes
```

_geom_hbond.angle_DHA (*float, su*)
_geom_hbond_angle_DHA (*cif_core.dic 2.0.1*)
The angle in degrees defined by the donor-, hydrogen- and acceptor-atom sites in a hydrogen bond.
The permitted range is $[0.0, \infty)$.
Related item: **_geom_hbond.angle_DHA_esd** (associated esd). **[geom_hbond]**

_geom_hbond.angle_DHA_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_geom_hbond.angle_DHA**.
Related item: **_geom_hbond.angle_DHA** (associated value). **[geom_hbond]**

_geom_hbond.atom_site_auth_asym_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_hbond.atom_site_auth_asym_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_hbond.atom_site_auth_asym_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_hbond.atom_site_auth_atom_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_hbond.atom_site_auth_atom_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_hbond.atom_site_auth_atom_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_hbond.atom_site_auth_comp_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.auth_comp_id in the ATOM_SITE category.

_geom_hbond.atom_site_auth_comp_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.auth_comp_id in the ATOM_SITE category.

_geom_hbond.atom_site_auth_comp_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.auth_comp_id in the ATOM_SITE category.

_geom_hbond.atom_site_auth_seq_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.auth_seq_id in the ATOM_SITE category.

_geom_hbond.atom_site_auth_seq_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.auth_seq_id in the ATOM_SITE category.

_geom_hbond.atom_site_auth_seq_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.auth_seq_id in the ATOM_SITE category.

* **_geom_hbond.atom_site_id_A**
_geom_hbond_atom_site_label_A (*cif_core.dic 2.0.1*)
The identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.id in the ATOM_SITE category.

* **_geom_hbond.atom_site_id_D**
_geom_hbond_atom_site_label_D (*cif_core.dic 2.0.1*)
The identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.id in the ATOM_SITE category.

* **_geom_hbond.atom_site_id_H**
_geom_hbond_atom_site_label_H (*cif_core.dic 2.0.1*)
The identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.id in the ATOM_SITE category.

_geom_hbond.atom_site_label_alt_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_alt_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_alt_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_alt_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_alt_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_alt_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_asym_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_asym_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_asym_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_asym_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_asym_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_asym_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_atom_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_atom_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_atom_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_atom_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_atom_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_atom_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_comp_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_comp_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_comp_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_comp_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_comp_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_comp_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_seq_id_A
An optional identifier of the acceptor-atom site that defines the hydrogen bond. This data item is a pointer to _atom_site.label_seq_id in the ATOM_SITE category.

_geom_hbond.atom_site_label_seq_id_D
An optional identifier of the donor-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_hbond.atom_site_label_seq_id_H
An optional identifier of the hydrogen-atom site that defines the hydrogen bond. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_hbond.dist_DA (*float, su*)
_geom_hbond_distance_DA (*cif_core.dic 2.0.1*)
The distance in ångströms between the donor- and acceptor-atom sites in a hydrogen bond.
The permitted range is $[0.0, \infty)$.
Related item: **_geom_hbond.dist_DA_esd** (associated esd). **[geom_hbond]**

_geom_hbond.dist_DA_esd (*float*)
The standard uncertainty (estimated standard deviation) in ångströms of **_geom_hbond.dist_DA**.
Related item: **_geom_hbond.dist_DH** (associated value). **[geom_hbond]**

_geom_hbond.dist_DH (*float, su*)
_geom_hbond_distance_DH (*cif_core.dic 2.0.1*)
The distance in ångströms between the donor- and hydrogen-atom sites in a hydrogen bond.
The permitted range is $[0.0, \infty)$.
Related item: **_geom_hbond.dist_DH_esd** (associated esd). **[geom_hbond]**

_geom_hbond.dist_DH_esd (*float*)
The standard uncertainty (estimated standard deviation) in ångströms of **_geom_hbond.dist_DH**.
Related item: **_geom_hbond.dist_DH** (associated value). **[geom_hbond]**

_geom_hbond.dist_HA (*float, su*)
_geom_hbond_distance_HA (*cif_core.dic 2.0.1*)
The distance in ångströms between the hydrogen- and acceptor-atom sites in a hydrogen bond.
The permitted range is $[0.0, \infty)$.
Related item: **_geom_hbond.dist_HA_esd** (associated esd). **[geom_hbond]**

_geom_hbond.dist_HA_esd (*float*)
The standard uncertainty (estimated standard deviation) in ångströms of **_geom_hbond.dist_HA**.
Related item: **_geom_hbond.dist_HA** (associated value). **[geom_hbond]**

_geom_hbond.publ_flag (*ucode*)
_geom_hbond_publ_flag (*cif_core.dic 2.0.1*)
This code signals whether the hydrogen-bond information is referred to in a publication or should be placed in a table of significant hydrogen-bond geometry.
The data value must be one of the following:

no	do not include bond in special list
n	abbreviation for 'no'
yes	do include bond in special list
y	abbreviation for 'yes'

[geom_hbond]

***_geom_hbond.site_symmetry_A** (*symop*)
_geom_hbond_site_symmetry_A (*cif_core.dic 2.0.1*)
The symmetry code of the acceptor-atom site that defines the hydrogen bond.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_hbond]**

***_geom_hbond.site_symmetry_D** (*symop*)
_geom_hbond_site_symmetry_D (*cif_core.dic 2.0.1*)
The symmetry code of the donor-atom site that defines the hydrogen bond.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_hbond]**

***_geom_hbond.site_symmetry_H** (*symop*)
_geom_hbond_site_symmetry_H (*cif_core.dic 2.0.1*)
The symmetry code of the hydrogen-atom site that defines the hydrogen bond.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_hbond]**

GEOM_TORSION

Data items in the GEOM_TORSION category record details about torsion angles as calculated from the contents of the ATOM, CELL and SYMMETRY data. The vector direction **_geom_torsion.atom_site_id_2** to **_geom_torsion.atom_site_id_3** is the viewing direction, and the torsion angle is the angle of twist required to superimpose the projection of the vector between site 2 and site 1 onto the projection of the vector between site 3 and site 4. Clockwise torsions are positive, anticlockwise torsions are negative.

Reference: Klyne, W. & Prelog, V. (1960). *Experientia*, **16**, 521–523.

Category group(s): **inclusive_group**
geom_group
Category key(s): **_geom_torsion.atom_site_id_1**
_geom_torsion.atom_site_id_2
_geom_torsion.atom_site_id_3
_geom_torsion.atom_site_id_4
_geom_torsion.site_symmetry_1
_geom_torsion.site_symmetry_2
_geom_torsion.site_symmetry_3
_geom_torsion.site_symmetry_4

*Example 1 – based on data set CLPHO6 of Ferguson, Ruhl, McKervey & Browne [Acta Cryst. (1992), C**48**, 2262–2264].*

```
loop_
_geom_torsion.atom_site_id_1
_geom_torsion.atom_site_id_2
_geom_torsion.atom_site_id_3
_geom_torsion.atom_site_id_4
_geom_torsion.value
_geom_torsion.site_symmetry_1
_geom_torsion.site_symmetry_2
_geom_torsion.site_symmetry_3
_geom_torsion.site_symmetry_4
_geom_torsion.publ_flag
 C(9)   O(2)    C(7)    C(2)     71.8  .  .  .  .      yes
 C(7)   O(2)    C(9)    C(10)  -168.0  .  .  .  2_666  yes
 C(10)  O(3)    C(8)    C(6)   -167.7  .  .  .  .      yes
 C(8)   O(3)    C(10)   C(9)    -69.7  .  .  .  2_666  yes
 O(1)   C(1)    C(2)    C(3)   -179.5  .  .  .  .      no
 O(1)   C(1)    C(2)    C(7)     -0.6  .  .  .  .      no
```

_geom_torsion.atom_site_auth_asym_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_asym_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_asym_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_asym_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_atom_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_atom_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_atom_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_atom_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_comp_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_comp_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_comp_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_comp_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_seq_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_seq_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_seq_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_geom_torsion.atom_site_auth_seq_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_geom_torsion.atom_site_id_1**
_geom_torsion_atom_site_label_1 (*cif_core.dic 2.0.1*)
The identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

* **_geom_torsion.atom_site_id_2**
_geom_torsion_atom_site_label_2 (*cif_core.dic 2.0.1*)
The identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

* **_geom_torsion.atom_site_id_3**
_geom_torsion_atom_site_label_3 (*cif_core.dic 2.0.1*)
The identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

* **_geom_torsion.atom_site_id_4**
_geom_torsion_atom_site_label_4 (*cif_core.dic 2.0.1*)
The identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_alt_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_alt_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_alt_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_alt_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_alt_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_asym_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_asym_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_asym_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_asym_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_atom_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_atom_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_atom_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_atom_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_comp_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_comp_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_comp_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_comp_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_seq_id_1
An optional identifier of the first of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_seq_id_2
An optional identifier of the second of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_seq_id_3
An optional identifier of the third of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_torsion.atom_site_label_seq_id_4
An optional identifier of the fourth of the four atom sites that define the torsion angle. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_geom_torsion.publ_flag (*ucode*)
_geom_torsion_publ_flag (*cif_core.dic 2.0.1*)
This code signals whether the torsion angle is referred to in a publication or should be placed in a table of significant torsion angles.
The data value must be one of the following:

no	do not include angle in special list
n	abbreviation for 'no'
yes	do include angle in special list
y	abbreviation for 'yes'

[geom_torsion]

***_geom_torsion.site_symmetry_1** (*symop*)
_geom_torsion_site_symmetry_1 (*cif_core.dic 2.0.1*)
The symmetry code of the first of the four atom sites that define the torsion angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_torsion]**

***_geom_torsion.site_symmetry_2** (*symop*)
_geom_torsion_site_symmetry_2 (*cif_core.dic 2.0.1*)
The symmetry code of the second of the four atom sites that define the torsion angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_torsion]**

***_geom_torsion.site_symmetry_3** (*symop*)
_geom_torsion_site_symmetry_3 (*cif_core.dic 2.0.1*)
The symmetry code of the third of the four atom sites that define the torsion angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_torsion]**

***_geom_torsion.site_symmetry_4** (*symop*)
_geom_torsion_site_symmetry_4 (*cif_core.dic 2.0.1*)
The symmetry code of the fourth of the four atom sites that define the torsion angle.
Where no value is given, the assumed value is '1_555'.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[geom_torsion]**

_geom_torsion.value (*float, su*)
_geom_torsion (*cif_core.dic 2.0.1*)
The value of the torsion angle in degrees.
Related item: **_geom_torsion.value_esd** (associated esd). **[geom_torsion]**

_geom_torsion.value_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_geom_torsion.value**.
Related item: **_geom_torsion.value** (associated value). **[geom_torsion]**

JOURNAL

Data items in the JOURNAL category record details about the book-keeping by the journal staff when processing a data block submitted for publication. The creator of a data block will not normally specify these data. The data names are not defined in the dictionary because they are for journal use only.
Category group(s): **inclusive_group**
iucr_group
Category key(s): **_journal.entry_id**

Example 1 – based on Acta Cryst. file for entry HL0007 [Willis, Beckwith & Tozer (1991). Acta Cryst. C47, 2276–2277].

```
_journal.entry_id                 'TOZ'
_journal.date_recd_electronic     1991-04-15
_journal.date_from_coeditor       1991-04-18
_journal.date_accepted            1991-04-18
_journal.date_printers_first      1991-08-07
_journal.date_proofs_out          1991-08-07
_journal.coeditor_code            HL0007
_journal.techeditor_code          C910963
_journal.coden_ASTM               ACSCEE
_journal.name_full      'Acta Crystallographica Section C'
_journal.year                     1991
_journal.volume                   47
_journal.issue                    NOV91
_journal.page_first               2276
_journal.page_last                2277
```

_journal.coden_ASTM (*line*)
_journal_coden_ASTM (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coden_Cambridge (*line*)
_journal_coden_Cambridge (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_address (*text*)
_journal_coeditor_address (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_code (*line*)
_journal_coeditor_code (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_email (*line*)
_journal_coeditor_email (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_fax (*line*)
_journal_coeditor_fax (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_name (*line*)
_journal_coeditor_name (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_notes (*text*)
_journal_coeditor_notes (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.coeditor_phone (*line*)
_journal_coeditor_phone (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.data_validation_number (*code*)
_journal_data_validation_number (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_accepted (*yyyy-mm-dd*)
_journal_date_accepted (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_from_coeditor (*yyyy-mm-dd*)
_journal_date_from_coeditor (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_printers_final (*yyyy-mm-dd*)
_journal_date_printers_final (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_printers_first (*yyyy-mm-dd*)
_journal_date_printers_first (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_proofs_in (*yyyy-mm-dd*)
_journal_date_proofs_in (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_proofs_out (*yyyy-mm-dd*)
_journal_date_proofs_out (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff. **[journal]**

_journal.date_recd_copyright (*yyyy-mm-dd*)
_journal_date_recd_copyright (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.date_recd_electronic (*yyyy-mm-dd*)
_journal_date_recd_electronic (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.date_recd_hard_copy (*yyyy-mm-dd*)
_journal_date_recd_hard_copy (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.date_to_coeditor (*yyyy-mm-dd*)
_journal_date_to_coeditor (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

* **_journal.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_journal.issue (*line*)
_journal_issue (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.language (*line*)
_journal_language (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.name_full (*line*)
_journal_name_full (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.page_first (*line*)
_journal_page_first (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.page_last (*line*)
_journal_page_last (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.paper_category (*line*)
_journal_paper_category (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.suppl_publ_number (*line*)
_journal_suppl_publ_number (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.suppl_publ_pages (*line*)
_journal_suppl_publ_pages (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_address (*text*)
_journal_techeditor_address (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_code (*line*)
_journal_techeditor_code (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_email (*line*)
_journal_techeditor_email (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_fax (*line*)
_journal_techeditor_fax (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_name (*line*)
_journal_techeditor_name (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_notes (*text*)
_journal_techeditor_notes (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.techeditor_phone (*line*)
_journal_techeditor_phone (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.volume (*line*)
_journal_volume (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

_journal.year (*line*)
_journal_year (*cif_core.dic 2.0.1*)
Journal data items are defined by the journal staff.
[journal]

JOURNAL_INDEX

Data items in the JOURNAL_INDEX category are used to list terms used to generate the journal indexes. The creator of a data block will not normally specify these data items.

Category group(s): **inclusive_group**
iucr_group
Category key(s): **_journal_index.type**
_journal_index.term

*Example 1 – based on a paper by Zhu, Reynolds, Klein & Trudell [Acta Cryst. (1994), **C50**, 2067–2069].*

```
loop_
_journal_index.type
_journal_index.term
_journal_index.subterm
  O   C16H19NO4            .
  S   alkaloids           (-)-norcocaine
  S   (-)-norcocaine       .
  S
;  [2R,3S-(2\b,3\b)]-methyl
   3-(benzoyloxy)-8-azabicyclo[3.2.1]octane-2-carboxylate
;
       .
```

_journal_index.subterm (*line*)
_journal_index_subterm (*cif_core.dic 2.0.1*)
Journal index data items are defined by the journal staff.
[journal_index]

_journal_index.term (*line*)
_journal_index_term (*cif_core.dic 2.0.1*)
Journal index data items are defined by the journal staff.
[journal_index]

_journal_index.type (*line*)
_journal_index_type (*cif_core.dic 2.0.1*)
Journal index data items are defined by the journal staff.

[journal_index]

PHASING

Data items in the PHASING category record details about the phasing of the structure, listing the various methods used in the phasing process. Details about the application of each method are listed in the appropriate subcategories.
Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing.method**

Example 1 – hypothetical example.

```
loop_
_phasing.method
'mir'
'averaging'
```

***_phasing.method** (*ucode*)
A listing of the method or methods used to phase this structure.
Examples: 'abinitio' (phasing by *ab initio* methods), 'averaging' (phase improvement by averaging over multiple images of the structure), 'dm' (phasing by direct methods), 'isas' (phasing by iterative single-wavelength anomalous scattering), 'isir' (phasing by iterative single-wavelength isomorphous replacement), 'isomorphous' (phasing beginning with phases calculated from an isomorphous structure), 'mad' (phasing by multiple-wavelength anomalous dispersion), 'mir' (phasing by multiple isomorphous replacement), 'miras' (phasing by multiple isomorphous replacement with anomalous scattering), 'mr' (phasing by molecular replacement), 'sir' (phasing by single isomorphous replacement), 'siras' (phasing by single isomorphous replacement with anomalous scattering). **[phasing]**

PHASING_AVERAGING

Data items in the PHASING_AVERAGING category record details about the phasing of the structure where methods involving averaging of multiple observations of the molecule in the asymmetric unit are involved.
Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_averaging.entry_id**

Example 1 – hypothetical example.

```
_phasing_averaging.entry_id    'EXAMHYPO'
_phasing_averaging.method
; Iterative threefold averaging alternating with phase
  extensions by 0.5 reciprocal lattice units per cycle.
;
_phasing_averaging.details
; The position of the threefold axis was redetermined every
  five cycles.
;
```

_phasing_averaging.details (*text*)
A description of special aspects of the averaging process.

[phasing_averaging]

***_phasing_averaging.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_phasing_averaging.method (*text*)
A description of the phase-averaging phasing method used to phase this structure. Note that this is not the computer program used, which is described in the SOFTWARE category, but rather the method itself. This data item should be used to describe significant methodological options used within the phase-averaging program.

[phasing_averaging]

PHASING_ISOMORPHOUS

Data items in the PHASING_ISOMORPHOUS category record details about the phasing of the structure where a model isomorphous to the structure being phased was used to generate the initial phases.
Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_isomorphous.entry_id**

Example 1 – based on PDB entry 4PHV and laboratory records for the structure corresponding to PDB entry 4PHV.

```
_phasing_isomorphous.parent           'PDB entry 5HVP'
_phasing_isomorphous.details
; The inhibitor and all solvent atoms were removed from the
  parent structure before beginning refinement. All static
  disorder present in the parent structure was also removed.
;
```

_phasing_isomorphous.details (*text*)
A description of special aspects of the isomorphous phasing.
Example:
```
; Residues 13-18 were eliminated from the starting model as it
was anticipated that binding of the inhibitor would cause a
structural rearrangement in this part of the structure.
;
```
[phasing_isomorphous]

***_phasing_isomorphous.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_phasing_isomorphous.method (*text*)
A description of the isomorphous-phasing method used to phase this structure. Note that this is not the computer program used, which is described in the SOFTWARE category, but rather the method itself. This data item should be used to describe significant methodological options used within the isomorphous phasing program.
Example:
```
; Iterative threefold averaging alternating with phase
extension by 0.5 reciprocal lattice units per cycle.
;
```
[phasing_isomorphous]

_phasing_isomorphous.parent (*text*)
Reference to the structure used to generate starting phases if the structure referenced in this data block was phased by virtue of being isomorphous to a known structure (*e.g.* a mutant that crystallizes in the same space group as the wild-type protein.)

[phasing_isomorphous]

PHASING_MAD

Data items in the PHASING_MAD category record details about the phasing of the structure where methods involving multiple-wavelength anomalous-dispersion techniques are involved.
Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MAD.entry_id**

*Example 1 – based on a paper by Shapiro et al. [Nature (London) (1995), **374**, 327–337].*

```
_phasing_MAD.entry_id       'NCAD'
```

_phasing_MAD.details (*text*)
A description of special aspects of the MAD phasing.

[phasing_MAD]

***_phasing_MAD.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_phasing_MAD.method (*text*)
A description of the MAD phasing method used to phase this structure. Note that this is not the computer program used, which is described in the SOFTWARE category, but rather the method itself. This data item should be used to describe significant methodological options used within the MAD phasing program.

[phasing_MAD]

PHASING_MAD_CLUST

Data items in the PHASING_MAD_CLUST category record details about a cluster of experiments that contributed to the generation of a set of phases.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MAD_clust.expt_id**
_phasing_MAD_clust.id

*Example 1 – based on a paper by Shapiro et al. [Nature (London) (1995), **374**, 327–337].*

```
loop_
_phasing_MAD_clust.id
_phasing_MAD_clust.expt_id
_phasing_MAD_clust.number_set
  '4 wavelength'  1  4
  '5 wavelength'  1  5
  '5 wavelength'  2  5
```

***_phasing_MAD_clust.expt_id**
This data item is a pointer to **_phasing_MAD_expt.id** in the PHASING_MAD_EXPT category.

***_phasing_MAD_clust.id** (*code*)
The value of **_phasing_MAD_clust.id** must, together with **_phasing_MAD_clust.expt_id**, uniquely identify a record in the PHASING_MAD_CLUST list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:
_phasing_MAD_set.clust_id,
_phasing_MAD_ratio.clust_id. [phasing_MAD_clust]

_phasing_MAD_clust.number_set (*int*)
The number of data sets in this cluster of data sets.

[phasing_MAD_clust]

PHASING_MAD_EXPT

Data items in the PHASING_MAD_EXPT category record details about a MAD phasing experiment, such as the number of experiments that were clustered together to produce a set of phases or the statistics for those phases.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MAD_expt.id**

*Example 1 – based on a paper by Shapiro et al. [Nature (London) (1995), **374**, 327–337].*

```
loop_
_phasing_MAD_expt.id
_phasing_MAD_expt.number_clust
_phasing_MAD_expt.R_normal_all
_phasing_MAD_expt.R_normal_anom_scat
_phasing_MAD_expt.delta_delta_phi
_phasing_MAD_expt.delta_phi_sigma
_phasing_MAD_expt.mean_fom
  1  2  0.063  0.451  58.5  20.3  0.88
  2  1  0.051  0.419  36.8  18.2  0.93
```

_phasing_MAD_expt.delta_delta_phi (*float*)
The difference between two independent determinations of **_phasing_MAD_expt.delta_phi**.

[phasing_MAD_expt]

_phasing_MAD_expt.delta_phi (*float*)
The phase difference between $F_t(h)$, the structure factor due to normal scattering from all atoms, and $F_a(h)$, the structure factor due to normal scattering from only the anomalous scatterers.

Related item: **_phasing_MAD_expt.delta_phi_sigma** (associated esd).

[phasing_MAD_expt]

_phasing_MAD_expt.delta_phi_sigma (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MAD_expt.delta_phi**.

Related item: **_phasing_MAD_expt.delta_phi** (associated value).

[phasing_MAD_expt]

***_phasing_MAD_expt.id** (*code*)
The value of **_phasing_MAD_expt.id** must uniquely identify each record in the PHASING_MAD_EXPT list.

The following item(s) have an equivalent role in their respective categories:
_phasing_MAD_clust.expt_id,
_phasing_MAD_set.expt_id,
_phasing_MAD_ratio.expt_id. [phasing_MAD_expt]

_phasing_MAD_expt.mean_fom (*float*)
The mean figure of merit.

[phasing_MAD_expt]

_phasing_MAD_expt.number_clust (*int*)
The number of clusters of data sets in this phasing experiment.

[phasing_MAD_expt]

_phasing_MAD_expt.R_normal_all (*float*)

[phasing_MAD_expt]

_phasing_MAD_expt.R_normal_anom_scat (*float*)

[phasing_MAD_expt]

PHASING_MAD_RATIO

Data items in the PHASING_MAD_RATIO category record the ratios of phasing statistics between pairs of data sets in a MAD phasing experiment, in given shells of resolution.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MAD_ratio.clust_id**
_phasing_MAD_ratio.expt_id
_phasing_MAD_ratio.wavelength_1
_phasing_MAD_ratio.wavelength_2

*Example 1 – based on a paper by Shapiro et al. [Nature (London) (1995), **374**, 327–337].*

```
loop_
_phasing_MAD_ratio.expt_id
_phasing_MAD_ratio.clust_id
_phasing_MAD_ratio.wavelength_1
_phasing_MAD_ratio.wavelength_2
_phasing_MAD_ratio.d_res_low
_phasing_MAD_ratio.d_res_high
_phasing_MAD_ratio.ratio_two_wl
_phasing_MAD_ratio.ratio_one_wl
_phasing_MAD_ratio.ratio_one_wl_centric
 1 '4 wavelength' 1.4013 1.4013 20.00 4.00     . 0.084 0.076
 1 '4 wavelength' 1.4013 1.3857 20.00 4.00 0.067     .     .
 1 '4 wavelength' 1.4013 1.3852 20.00 4.00 0.051     .     .
 1 '4 wavelength' 1.4013 1.3847 20.00 4.00 0.044     .     .
 1 '4 wavelength' 1.3857 1.3857 20.00 4.00     . 0.110 0.049
 1 '4 wavelength' 1.3857 1.3852 20.00 4.00 0.049     .     .
 1 '4 wavelength' 1.3857 1.3847 20.00 4.00 0.067     .     .
 1 '4 wavelength' 1.3852 1.3852 20.00 4.00     . 0.149 0.072
 1 '4 wavelength' 1.3852 1.3847 20.00 4.00 0.039     .     .
 1 '4 wavelength' 1.3847 1.3847 20.00 4.00     . 0.102 0.071

 1 '4 wavelength' 1.4013 1.4013  4.00 3.00     . 0.114 0.111
 1 '4 wavelength' 1.4013 1.3857  4.00 3.00 0.089     .     .
 1 '4 wavelength' 1.4013 1.3852  4.00 3.00 0.086     .     .
 1 '4 wavelength' 1.4013 1.3847  4.00 3.00 0.077     .     .
 1 '4 wavelength' 1.3857 1.3857  4.00 3.00     . 0.140 0.127
 1 '4 wavelength' 1.3857 1.3852  4.00 3.00 0.085     .     .
 1 '4 wavelength' 1.3857 1.3847  4.00 3.00 0.089     .     .
 1 '4 wavelength' 1.3852 1.3852  4.00 3.00     . 0.155 0.119
 1 '4 wavelength' 1.3852 1.3847  4.00 3.00 0.082     .     .
 1 '4 wavelength' 1.3847 1.3847  4.00 3.00     . 0.124 0.120

 1 '5 wavelength' 1.3857 1.3857 20.00 4.00     . 0.075 0.027
 1 '5 wavelength' 1.3857 1.3852 20.00 4.00 0.041     .     .
 1 '5 wavelength' 1.3857 1.3847 20.00 4.00 0.060     .     .
 1 '5 wavelength' 1.3857 1.3784 20.00 4.00 0.057     .     .
 1 '5 wavelength' 1.3857 1.2862 20.00 4.00 0.072     .     .
 1 '5 wavelength' 1.3852 1.3852 20.00 4.00     . 0.105 0.032
 1 '5 wavelength' 1.3852 1.3847 20.00 4.00 0.036     .     .
 1 '5 wavelength' 1.3852 1.3784 20.00 4.00 0.044     .     .
 1 '5 wavelength' 1.3852 1.2862 20.00 4.00 0.065     .     .
 1 '5 wavelength' 1.3847 1.3847 20.00 4.00     . 0.072 0.031
 1 '5 wavelength' 1.3847 1.3784 20.00 4.00 0.040     .     .
 1 '5 wavelength' 1.3847 1.2862 20.00 4.00 0.059     .     .
 1 '5 wavelength' 1.3784 1.3784 20.00 4.00     . 0.059 0.032
 1 '5 wavelength' 1.3784 1.2862 20.00 4.00 0.059     .     .
 1 '5 wavelength' 1.2862 1.3847 20.00 4.00     . 0.058 0.028

 1 '5 wavelength' 1.3857 1.3857  4.00 3.00     . 0.078 0.075
 1 '5 wavelength' 1.3857 1.3852  4.00 3.00 0.059     .     .
 1 '5 wavelength' 1.3857 1.3847  4.00 3.00 0.067     .     .
 1 '5 wavelength' 1.3857 1.3784  4.00 3.00 0.084     .     .
 1 '5 wavelength' 1.3857 1.2862  4.00 3.00 0.073     .     .
 1 '5 wavelength' 1.3852 1.3852  4.00 3.00     . 0.101 0.088
 1 '5 wavelength' 1.3852 1.3847  4.00 3.00 0.066     .     .
 1 '5 wavelength' 1.3852 1.3784  4.00 3.00 0.082     .     .
 1 '5 wavelength' 1.3852 1.2862  4.00 3.00 0.085     .     .
 1 '5 wavelength' 1.3847 1.3847  4.00 3.00     . 0.097 0.074
 1 '5 wavelength' 1.3847 1.3784  4.00 3.00 0.081     .     .
 1 '5 wavelength' 1.3847 1.2862  4.00 3.00 0.085     .     .
 1 '5 wavelength' 1.3784 1.3784  4.00 3.00     . 0.114 0.089
 1 '5 wavelength' 1.3784 1.2862  4.00 3.00 0.103     .     .
 1 '5 wavelength' 1.2862 1.2862  4.00 3.00     . 0.062 0.060

 2 '5 wavelength' 0.7263 0.7263 15.00 3.00     . 0.035 0.026
 2 '5 wavelength' 0.7263 0.7251 15.00 3.00 0.028     .     .
 2 '5 wavelength' 0.7263 0.7284 15.00 3.00 0.023     .     .
 2 '5 wavelength' 0.7263 0.7246 15.00 3.00 0.025     .     .
 2 '5 wavelength' 0.7263 0.7217 15.00 3.00 0.026     .     .
 2 '5 wavelength' 0.7251 0.7251 15.00 3.00     . 0.060 0.026
 2 '5 wavelength' 0.7251 0.7284 15.00 3.00 0.029     .     .
 2 '5 wavelength' 0.7251 0.7246 15.00 3.00 0.031     .     .
 2 '5 wavelength' 0.7251 0.7217 15.00 3.00 0.035     .     .
 2 '5 wavelength' 0.7284 0.7284 15.00 3.00     . 0.075 0.030
 2 '5 wavelength' 0.7284 0.7246 15.00 3.00 0.023     .     .
 2 '5 wavelength' 0.7284 0.7217 15.00 3.00 0.027     .     .
 2 '5 wavelength' 0.7246 0.7246 15.00 3.00     . 0.069 0.026
 2 '5 wavelength' 0.7246 0.7217 15.00 3.00 0.024     .     .
 2 '5 wavelength' 0.7217 0.7284 15.00 3.00     . 0.060 0.028

 2 '5 wavelength' 0.7263 0.7263  3.00 1.90     . 0.060 0.050
 2 '5 wavelength' 0.7263 0.7251  3.00 1.90 0.056     .     .
 2 '5 wavelength' 0.7263 0.7284  3.00 1.90 0.055     .     .
 2 '5 wavelength' 0.7263 0.7246  3.00 1.90 0.053     .     .
 2 '5 wavelength' 0.7263 0.7217  3.00 1.90 0.056     .     .
 2 '5 wavelength' 0.7251 0.7251  3.00 1.90     . 0.089 0.050
 2 '5 wavelength' 0.7251 0.7284  3.00 1.90 0.054     .     .
 2 '5 wavelength' 0.7251 0.7246  3.00 1.90 0.058     .     .
 2 '5 wavelength' 0.7251 0.7217  3.00 1.90 0.063     .     .
 2 '5 wavelength' 0.7284 0.7284  3.00 1.90     . 0.104 0.057
 2 '5 wavelength' 0.7284 0.7246  3.00 1.90 0.052     .     .
 2 '5 wavelength' 0.7284 0.7217  3.00 1.90 0.057     .     .
 2 '5 wavelength' 0.7246 0.7246  3.00 1.90     . 0.098 0.052
 2 '5 wavelength' 0.7246 0.7217  3.00 1.90 0.054     .     .
 2 '5 wavelength' 0.7217 0.7284  3.00 1.90     . 0.089 0.060
```

* **_phasing_MAD_ratio.clust_id**

This data item is a pointer to **_phasing_MAD_clust.id** in the PHASING_MAD_CLUST category.

_phasing_MAD_ratio.d_res_high (*float*)

The lowest value for the interplanar spacings for the reflection data used for the comparison of Bijvoet differences. This is called the highest resolution.

[phasing_MAD_ratio]

_phasing_MAD_ratio.d_res_low (*float*)

The highest value for the interplanar spacings for the reflection data used for the comparison of Bijvoet differences. This is called the lowest resolution.

[phasing_MAD_ratio]

* **_phasing_MAD_ratio.expt_id**

This data item is a pointer to **_phasing_MAD_expt.id** in the PHASING_MAD_EXPT category.

_phasing_MAD_ratio.ratio_one_wl (*float*)

The root-mean-square Bijvoet difference at one wavelength for all reflections.

[phasing_MAD_ratio]

_phasing_MAD_ratio.ratio_one_wl_centric (*float*)

The root-mean-square Bijvoet difference at one wavelength for centric reflections. This would be equal to zero for perfect data and thus serves as an estimate of the noise in the anomalous signals.

[phasing_MAD_ratio]

_phasing_MAD_ratio.ratio_two_wl (*float*)

The root-mean-square dispersive Bijvoet difference between two wavelengths for all reflections.

[phasing_MAD_ratio]

* **_phasing_MAD_ratio.wavelength_1**

This data item is a pointer to **_phasing_MAD_set.wavelength** in the PHASING_MAD_SET category.

* **_phasing_MAD_ratio.wavelength_2**

This data item is a pointer to **_phasing_MAD_set.wavelength** in the PHASING_MAD_SET category.

PHASING_MAD_SET

Data items in the PHASING_MAD_SET category record details about the individual data sets used in a MAD phasing experiment.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MAD_set.expt_id**
_phasing_MAD_set.clust_id
_phasing_MAD_set.set_id
_phasing_MAD_set.wavelength

*Example 1 – based on a paper by Shapiro et al. [Nature (London) (1995), **374**, 327–337].*

```
loop_
_phasing_MAD_set.expt_id
_phasing_MAD_set.clust_id
_phasing_MAD_set.set_id
_phasing_MAD_set.wavelength
_phasing_MAD_set.wavelength_details
_phasing_MAD_set.d_res_low
_phasing_MAD_set.d_res_high
_phasing_MAD_set.f_prime
_phasing_MAD_set.f_double_prime
1 '4 wavelength' aa 1.4013 'pre-edge'        20.00   3.00
     -12.48   3.80
1 '4 wavelength' bb 1.3857 'peak'           20.00   3.00
     -31.22  17.20
1 '4 wavelength' cc 1.3852 'edge'            20.00   3.00
     -13.97  29.17
1 '4 wavelength' dd 1.3847 'remote'          20.00   3.00
     -6.67  17.34
1 '5 wavelength' ee 1.3857 'ascending edge'  20.00   3.00
     -28.33  14.84
1 '5 wavelength' ff 1.3852 'peak'            20.00   3.00
     -21.50  30.23
1 '5 wavelength' gg 1.3847 'descending edge' 20.00   3.00
     -10.71  20.35
1 '5 wavelength' hh 1.3784 'remote 1'        20.00   3.00
     -14.45  11.84
1 '5 wavelength' ii 1.2862 'remote 2'        20.00   3.00
     -9.03   9.01
2 '5 wavelength' jj 0.7263 'pre-edge'        15.00   1.90
     -21.10   4.08
2 '5 wavelength' kk 0.7251 'edge'            15.00   1.90
     -34.72   7.92
2 '5 wavelength' ll 0.7248 'peak'            15.00   1.90
     -24.87  10.30
2 '5 wavelength' mm 0.7246 'descending edge' 15.00   1.90
     -17.43   9.62
2 '5 wavelength' nn 0.7217 'remote'          15.00   1.90
     -13.26   8.40
```

* **_phasing_MAD_set.clust_id**
This data item is a pointer to **_phasing_MAD_clust.id** in the PHASING_MAD_CLUST category.

_phasing_MAD_set.d_res_high (*float*)
The lowest value for the interplanar spacings for the reflection data used for this set of data. This is called the highest resolution.
[phasing_MAD_set]

_phasing_MAD_set.d_res_low (*float*)
The highest value for the interplanar spacings for the reflection data used for this set of data. This is called the lowest resolution.
[phasing_MAD_set]

* **_phasing_MAD_set.expt_id**
This data item is a pointer to **_phasing_MAD_expt.id** in the PHASING_MAD_EXPT category.

_phasing_MAD_set.f_double_prime (*float*)
The f'' component of the anomalous scattering factor for this wavelength.
[phasing_MAD_set]

_phasing_MAD_set.f_prime (*float*)
The f' component of the anomalous scattering factor for this wavelength.
[phasing_MAD_set]

* **_phasing_MAD_set.set_id**
This data item is a pointer to **_phasing_set.id** in the PHASING_SET category.

* **_phasing_MAD_set.wavelength** (*float*)
The wavelength at which this data set was measured.
The following item(s) have an equivalent role in their respective categories:
_phasing_MAD_ratio.wavelength_1,
_phasing_MAD_ratio.wavelength_2. **[phasing_MAD_set]**

_phasing_MAD_set.wavelength_details (*text*)
A descriptor for this wavelength in this cluster of data sets.
Examples: 'peak', 'remote', 'ascending edge'. **[phasing_MAD_set]**

PHASING_MIR

Data items in the PHASING_MIR category record details about the phasing of the structure where methods involving isomorphous replacement are involved. All isomorphous-replacement-based techniques are covered by this category, including single isomorphous replacement (SIR), multiple isomorphous replacement (MIR) and single or multiple isomorphous replacement plus anomalous scattering (SIRAS, MIRAS).

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MIR.entry_id**

*Example 1 – based on a paper by Zanotti et al. [J. Biol. Chem. (1993), **268**, 10728–10738].*

```
_phasing_MIR.method
; Standard phase refinement (Blow & Crick, 1959)
;
```

* **_phasing_MIR.d_res_high** (*float*)
_phasing_MIR.ebi_d_res_high (*ebi_extensions 1.0*)
The lowest value in ångströms for the interplanar spacings for the reflection data used for the native data set. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[phasing_MIR]**

* **_phasing_MIR.d_res_low** (*float*)
_phasing_MIR.ebi_d_res_low (*ebi_extensions 1.0*)
The highest value in ångströms for the interplanar spacings for the reflection data used for the native data set. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[phasing_MIR]**

_phasing_MIR.details (*text*)
A description of special aspects of the isomorphous-replacement phasing.
[phasing_MIR]

* **_phasing_MIR.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_phasing_MIR.FOM (*float*)
_phasing_MIR.ebi_fom (*ebi_extensions 1.0*)
The mean value of the figure of merit m for all reflections phased in the native data set.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha \, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct and the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR]**

_phasing_MIR.FOM_acentric (*float*)
_phasing_MIR.ebi_fom_acentric (*ebi_extensions 1.0*)
The mean value of the figure of merit m for the acentric reflections phased in the native data set.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha \, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct and the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR]**

_phasing_MIR.FOM_centric (*float*)
_phasing_MIR.ebi_fom_centric (*ebi_extensions 1.0*)
The mean value of the figure of merit m for the centric reflections phased in the native data set.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha \, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct and the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR]**

_phasing_MIR.method (*text*)
A description of the MIR phasing method applied to phase this structure. Note that this is not the computer program used, which is described in the SOFTWARE category, but rather the method itself. This data item should be used to describe significant methodological options used within the MIR phasing program.
[phasing_MIR]

_phasing_MIR.reflns (*int*)
_phasing_MIR.ebi_reflns (*ebi_extensions 1.0*)
The total number of reflections phased in the native data set.
The permitted range is $[0, \infty)$. **[phasing_MIR]**

_phasing_MIR.reflns_acentric (*int*)
_phasing_MIR.ebi_reflns_acentric (*ebi_extensions 1.0*)
The number of acentric reflections phased in the native data set.
The permitted range is $[0, \infty)$. **[phasing_MIR]**

_phasing_MIR.reflns_centric (*int*)
_phasing_MIR.ebi_reflns_centric (*ebi_extensions 1.0*)
The number of centric reflections phased in the native data set.
The permitted range is $[0, \infty)$. **[phasing_MIR]**

_phasing_MIR.reflns_criterion (*text*)
_phasing_MIR.ebi_reflns_criteria (*ebi_extensions 1.0*)
Criterion used to limit the reflections used in the phasing calculations.
Example: '`> 4 \s(I)`'. **[phasing_MIR]**

PHASING_MIR_DER

Data items in the PHASING_MIR_DER category record details about individual derivatives used in the phasing of the structure when methods involving isomorphous replacement are involved. A derivative in this context does not necessarily equate with a data set; for instance, the same data set could be used to one resolution limit as an isomorphous scatterer and to a different resolution (and with a different σ cut-off) as an anomalous scatterer. These would be treated as two distinct derivatives, although both derivatives would point to the same data sets *via* **_phasing_MIR_der.der_set_id** and **_phasing_MIR_der.native_set_id**.
Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MIR_der.id**

*Example 1 – based on a paper by Zanotti et al. [J. Biol. Chem. (1993), **268**, 10728–10738].*

```
loop_
_phasing_MIR_der.id
_phasing_MIR_der.number_of_sites
_phasing_MIR_der.details
KAu(CN)2  3  'major site interpreted in difference Patterson'
K2HgI4    6  'sites found in cross-difference Fourier'
K3IrCl6   2  'sites found in cross-difference Fourier'
All      11  'data for all three derivatives combined'
```

***_phasing_MIR_der.d_res_high** (*float*)
The lowest value for the interplanar spacings for the reflection data used for this derivative. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

***_phasing_MIR_der.d_res_low** (*float*)
The highest value for the interplanar spacings for the reflection data used for this derivative. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

***_phasing_MIR_der.der_set_id**
The data set that was treated as the derivative in this experiment. This data item is a pointer to **_phasing_set.id** in the PHASING_SET category.

_phasing_MIR_der.details (*text*)
A description of special aspects of this derivative, its data, its solution or its use in phasing.
[phasing_MIR_der]

***_phasing_MIR_der.id** (*line*)
The value of **_phasing_MIR_der.id** must uniquely identify a record in the PHASING_MIR_DER list. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_phasing_MIR_der_refln.der_id,
_phasing_MIR_der_shell.der_id,
_phasing_MIR_der_site.der_id.
Examples: '`KAu(CN)2`', '`K2HgI4_anom`', '`K2HgI4_iso`'. **[phasing_MIR_der]**

***_phasing_MIR_der.native_set_id**
The data set that was treated as the native in this experiment. This data item is a pointer to **_phasing_set.id** in the PHASING_SET category.

_phasing_MIR_der.number_of_sites (*int*)
The number of heavy-atom sites in this derivative.
[phasing_MIR_der]

_phasing_MIR_der.power_acentric (*float*)
_phasing_MIR_der.ebi_power_acentric (*ebi_extensions 1.0*)
The mean phasing power P for acentric reflections for this derivative.

$$P = \left(\frac{\sum |F(h)^2_{\rm calc}|}{\sum |F(ph)_{\rm obs} - F(ph)_{\rm calc}|^2} \right)^{1/2},$$

where $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of this derivative, $F(ph)_{\rm calc}$ = the calculated structure-factor amplitude of this derivative and $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.power_centric (*float*)
_phasing_MIR_der.ebi_power_centric (*ebi_extensions 1.0*)
The mean phasing power P for centric reflections for this derivative.

$$P = \left(\frac{\sum |F(h)^2_{\rm calc}|}{\sum |F(ph)_{\rm obs} - F(ph)_{\rm calc}|^2} \right)^{1/2},$$

where $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of this derivative, $F(ph)_{\rm calc}$ = the calculated structure-factor amplitude of this derivative and $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.R_cullis_acentric (*float*)
_phasing_MIR_der.ebi_Rcullis_acentric (*ebi_extensions 1.0*)
Residual factor $R_{\rm cullis,acen}$ for acentric reflections for this derivative. The Cullis R factor was originally defined only for centric reflections. It is, however, also a useful statistical measure for acentric reflections, which is how it is used in this data item.

$$R_{\rm cullis,acen} = \frac{\sum \Big| |F(ph)_{\rm obs} \pm F(p)_{\rm obs}| - F(h)_{\rm calc} \Big|}{\sum |F(ph)_{\rm obs} - F(p)_{\rm obs}|},$$

where $F(p)_{\rm obs}$ = the observed structure-factor amplitude of the native, $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of the derivative and $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.

Reference: Cullis, A. F., Muirhead, H., Perutz, M. F., Rossmann, M. G. & North, A. C. T. (1961). *Proc. R. Soc. London Ser. A*, **265**, 15–38.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.R_cullis_anomalous (*float*)
_phasing_MIR_der.ebi_Rcullis_anomalous (*ebi_extensions 1.0*)
Residual factor $R_{\rm cullis,ano}$ for anomalous reflections for this derivative. The Cullis R factor was originally defined only for centric reflections. It is, however, also a useful statistical measure for anomalous reflections, which is how it is used in this data item. This is tabulated for acentric terms. A value less than 1.0 means there is some contribution to the phasing from the anomalous data.

$$R_{\rm cullis,ano} = \frac{\sum |F(ph+)_{\rm obs} F(ph-)_{\rm obs} - F(h+)_{\rm calc} - F(h-)_{\rm calc}|}{\sum |F(ph+)_{\rm obs} - F(ph-)_{\rm obs}|},$$

where $F(ph+)_{\rm obs}$ = the observed positive Friedel structure-factor amplitude for the derivative, $F(ph-)_{\rm obs}$ = the observed negative Friedel structure-factor amplitude for the derivative, $F(h+)_{\rm calc}$ = the calculated positive Friedel structure-factor amplitude from the heavy-atom model and $F(h-)_{\rm calc}$ = the calculated negative Friedel structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.

Reference: Cullis, A. F., Muirhead, H., Perutz, M. F., Rossmann, M. G. & North, A. C. T. (1961). *Proc. R. Soc. London Ser. A*, **265**, 15–38.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.R_cullis_centric (*float*)
_phasing_MIR_der.ebi_Rcullis_centric (*ebi_extensions 1.0*)
Residual factor $R_{\rm cullis}$ for centric reflections for this derivative.

$$R_{\rm cullis} = \frac{\sum \Big| |F(ph)_{\rm obs} \pm F(p)_{\rm obs}| - F(h)_{\rm calc} \Big|}{\sum |F(ph)_{\rm obs} - F(p)_{\rm obs}|},$$

where $F(p)_{\rm obs}$ = the observed structure-factor amplitude of the native, $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of the derivative and $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.

Reference: Cullis, A. F., Muirhead, H., Perutz, M. F., Rossmann, M. G. & North, A. C. T. (1961). *Proc. R. Soc. London Ser. A*, **265**, 15–38.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.reflns_acentric (*int*)
_phasing_MIR_der.ebi_reflns_acentric (*ebi_extensions 1.0*)
The number of acentric reflections used in phasing for this derivative.
The permitted range is $[0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.reflns_anomalous (*int*)
_phasing_MIR_der.ebi_reflns_anomalous (*ebi_extensions 1.0*)
The number of anomalous reflections used in phasing for this derivative.
The permitted range is $[0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.reflns_centric (*int*)
_phasing_MIR_der.ebi_reflns_centric (*ebi_extensions 1.0*)
The number of centric reflections used in phasing for this derivative.
The permitted range is $[0, \infty)$. **[phasing_MIR_der]**

_phasing_MIR_der.reflns_criteria (*text*)
Criteria used to limit the reflections used in the phasing calculations.
Example: '`> 4 \s(I)`'. **[phasing_MIR_der]**

PHASING_MIR_DER_REFLN

Data items in the PHASING_MIR_DER_REFLN category record details about the calculated structure factors obtained in an MIR phasing experiment. This list may contain information from a number of different derivatives; **_phasing_MIR_der_refln.der_id** indicates to which derivative a given record corresponds. (A derivative in this context does not necessarily equate with a data set; see the definition of the PHASING_MIR_DER category for a discussion of the meaning of 'derivative'.) It is not necessary for the data items describing the measured value of F to appear in this list, as they will be given in the PHASING_SET_REFLN category. However, these items can also be listed here for completeness.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MIR_der_refln.index_h**
_phasing_MIR_der_refln.index_k
_phasing_MIR_der_refln.index_l
_phasing_MIR_der_refln.der_id
_phasing_MIR_der_refln.set_id

Example 1 – based on laboratory records for the 6,1,25 reflection of an Hg/Pt derivative of protein NS1.

```
_phasing_MIR_der_refln.index_h         6
_phasing_MIR_der_refln.index_k         1
_phasing_MIR_der_refln.index_l        25
_phasing_MIR_der_refln.der_id         HGPT1
_phasing_MIR_der_refln.set_id       'NS1-96'
_phasing_MIR_der_refln.F_calc_au     106.66
_phasing_MIR_der_refln.F_meas_au     204.67
_phasing_MIR_der_refln.F_meas_sigma    6.21
_phasing_MIR_der_refln.HL_A_iso       -3.15
_phasing_MIR_der_refln.HL_B_iso       -0.76
_phasing_MIR_der_refln.HL_C_iso        0.65
_phasing_MIR_der_refln.HL_D_iso        0.23
_phasing_MIR_der_refln.phase_calc    194.48
```

* **_phasing_MIR_der_refln.der_id**
This data item is a pointer to **_phasing_MIR_der.id** in the PHASING_MIR_DER category.

_phasing_MIR_der_refln.F_calc (*float*)
The calculated value of the structure factor for this derivative, in electrons.
Related item: **_phasing_MIR_der_refln.F_calc_au** (conversion arbitrary).
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.F_calc_au (*float*)
The calculated value of the structure factor for this derivative, in arbitrary units.
Related item: **_phasing_MIR_der_refln.F_calc** (conversion arbitrary).
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.F_meas (*float, su*)
The measured value of the structure factor for this derivative, in electrons.
Related items: **_phasing_MIR_der_refln.F_meas_sigma** (associated esd), **_phasing_MIR_der_refln.F_meas_au** (conversion arbitrary).
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.F_meas_au (*float, su*)
The measured value of the structure factor for this derivative, in arbitrary units.
Related items: **_phasing_MIR_der_refln.F_meas_sigma_au** (associated esd), **_phasing_MIR_der_refln.F_meas** (conversion arbitrary).
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.F_meas_sigma (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_refln.F_meas**, in electrons.
Related items: **_phasing_MIR_der_refln.F_meas** (associated value), **_phasing_MIR_der_refln.F_meas_sigma_au** (conversion arbitrary).
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.F_meas_sigma_au (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_refln.F_meas_au**, in arbitrary units.
Related items: **_phasing_MIR_der_refln.F_meas_au** (associated value), **_phasing_MIR_der_refln.F_meas_sigma** (conversion arbitrary).
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.HL_A_iso (*float*)
The isomorphous Hendrickson–Lattman coefficient $A_{\rm iso}$ for this reflection for this derivative.

$$A_{\rm iso} = -\frac{2.0[F(p)^2_{\rm obs} + F(h)^2_{\rm calc} - F(ph)^2_{\rm obs}]F(p)_{\rm obs}\cos(\alpha_{h\rm calc})}{E^2},$$

where $E = [F(ph)_{\rm obs} - F(p)_{\rm obs} - F(h)_{\rm calc}]^2$ for centric reflections, $E = \{2^{1/2}[F(ph)_{\rm obs} - F(p)_{\rm obs}] - F(h)_{\rm calc}\}^2$ for acentric reflections, $F(p)_{\rm obs}$ = the observed structure-factor amplitude of the native, $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of the derivative, $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model and $\alpha_{h\rm calc}$ = the calculated phase from the heavy-atom model.

This coefficient appears in the expression for the phase probability of each isomorphous derivative:

$$P_i(\alpha) = \exp(k + A\cos\alpha + B\sin\alpha + C\cos 2\alpha + D\sin 2\alpha).$$

Reference: Hendrickson, W. A. & Lattman, E. E. (1970). *Acta Cryst.* B**26**, 136–143.
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.HL_B_iso (*float*)
The isomorphous Hendrickson–Lattman coefficient $B_{\rm iso}$ for this reflection for this derivative.

$$B_{\rm iso} = -\frac{2.0[F(p)^2_{\rm obs} + F(h)^2_{\rm calc} - F(ph)^2_{\rm obs}]F(p)_{\rm obs}\sin(\alpha_{h\rm calc})}{E^2},$$

where $E = [F(ph)_{\rm obs} - F(p)_{\rm obs} - F(h)_{\rm calc}]^2$ for centric reflections, $E = \{2^{1/2}[F(ph)_{\rm obs} - F(p)_{\rm obs}] - F(h)_{\rm calc}\}^2$ for acentric reflections, $F(p)_{\rm obs}$ = the observed structure-factor amplitude of the native, $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of the derivative, $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model and $\alpha_{h\rm calc}$ = the calculated phase from the heavy-atom model.

This coefficient appears in the expression for the phase probability of each isomorphous derivative:

$$P_i(\alpha) = \exp(k + A\cos\alpha + B\sin\alpha + C\cos 2\alpha + D\sin 2\alpha).$$

Reference: Hendrickson, W. A. & Lattman, E. E. (1970). *Acta Cryst.* B**26**, 136–143.
[phasing_MIR_der_refln]

_phasing_MIR_der_refln.HL_C_iso (*float*)
The isomorphous Hendrickson–Lattman coefficient $C_{\rm iso}$ for this reflection for this derivative.

$$C_{\rm iso} = -\frac{F(p)^2_{\rm obs}[\sin(\alpha_{h\rm calc})^2 - \cos(\alpha_{h\rm calc})^2]}{E^2},$$

where $E = [F(ph)_{\rm obs} - F(p)_{\rm obs} - F(h)_{\rm calc}]^2$ for centric reflections, $E = \{2^{1/2}[F(ph)_{\rm obs} - F(p)_{\rm obs}] - F(h)_{\rm calc}\}^2$ for acentric reflections, $F(p)_{\rm obs}$ = the observed structure-factor amplitude of the native, $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of the derivative, $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model and $\alpha_{h\rm calc}$ = the calculated phase from the heavy-atom model.

This coefficient appears in the expression for the phase probability of each isomorphous derivative:

$$P_i(\alpha) = \exp(k + A\cos\alpha + B\sin\alpha + C\cos 2\alpha + D\sin 2\alpha).$$

Reference: Hendrickson, W. A. & Lattman, E. E. (1970). *Acta Cryst.* B**26**, 136–143.

[phasing_MIR_der_refln]

_phasing_MIR_der_refln.HL_D_iso (*float*)
The isomorphous Hendrickson–Lattman coefficient $D_{\rm iso}$ for this reflection for this derivative.

$$D_{\rm iso} = -\frac{2.0F(p)^2_{\rm obs}\sin(\alpha_{h\rm calc})^2 + \cos(\alpha_{h\rm calc})^2}{E^2},$$

where $E = [F(ph)_{\rm obs} - F(p)_{\rm obs} - F(h)_{\rm calc}]^2$ for centric reflections, $E = \{2^{1/2}[F(ph)_{\rm obs} - F(p)_{\rm obs}] - F(h)_{\rm calc}\}^2$ for acentric reflections, $F(p)_{\rm obs}$ = the observed structure-factor amplitude of the native, $F(ph)_{\rm obs}$ = the observed structure-factor amplitude of the derivative, $F(h)_{\rm calc}$ = the calculated structure-factor amplitude from the heavy-atom model and $\alpha_{h\rm calc}$ = the calculated phase from the heavy-atom model.

This coefficient appears in the expression for the phase probability of each isomorphous derivative:

$$P_i(\alpha) = \exp(k + A\cos\alpha + B\sin\alpha + C\cos 2\alpha + D\sin 2\alpha).$$

Reference: Hendrickson, W. A. & Lattman, E. E. (1970). *Acta Cryst.* B**26**, 136–143.

[phasing_MIR_der_refln]

* **_phasing_MIR_der_refln.index_h** (*int*)
Miller index h for this reflection for this derivative.

[phasing_MIR_der_refln]

* **_phasing_MIR_der_refln.index_k** (*int*)
Miller index k for this reflection for this derivative.

[phasing_MIR_der_refln]

* **_phasing_MIR_der_refln.index_l** (*int*)
Miller index l for this reflection for this derivative.

[phasing_MIR_der_refln]

_phasing_MIR_der_refln.phase_calc (*float*)
The calculated value of the structure-factor phase based on the heavy-atom model for this derivative in degrees.

[phasing_MIR_der_refln]

* **_phasing_MIR_der_refln.set_id**
This data item is a pointer to **_phasing_set.id** in the PHASING_SET category.

PHASING_MIR_DER_SHELL

Data items in the PHASING_MIR_DER_SHELL category record statistics, broken down into shells of resolution, for an MIR phasing experiment. This list may contain information from a number of different derivatives; **_phasing_MIR_der_shell.der_id** indicates to which derivative a given record corresponds. (A derivative in this context does not necessarily equate with a data set; see the definition of the PHASING_MIR_DER category for a discussion of the meaning of 'derivative'.)

Category group(s): **inclusive_group**
phasing_group

Category key(s): **_phasing_MIR_der_shell.der_id**
_phasing_MIR_der_shell.d_res_low
_phasing_MIR_der_shell.d_res_high

*Example 1 – based on a paper by Zanotti et al. [J. Biol. Chem. (1993), **268**, 10728–10738] with addition of an arbitrary low-resolution limit.*

```
loop_
_phasing_MIR_der_shell.der_id
_phasing_MIR_der_shell.d_res_low
_phasing_MIR_der_shell.d_res_high
_phasing_MIR_der_shell.ha_ampl
_phasing_MIR_der_shell.loc
 KAu(CN)2  15.0  8.3   54   26
 KAu(CN)2   8.3  6.4   54   20
 KAu(CN)2   6.4  5.2   50   20
 KAu(CN)2   5.2  4.4   44   23
 KAu(CN)2   4.4  3.8   39   23
 KAu(CN)2   3.8  3.4   33   21
 KAu(CN)2   3.4  3.0   28   17
 KAu(CN)2  15.0  3.0   38   21
 K2HgI4    15.0  8.3  149   87
 K2HgI4     8.3  6.4  121   73
 K2HgI4     6.4  5.2   95   61
 K2HgI4     5.2  4.4   80   60
 K2HgI4     4.4  3.8   73   63
 K2HgI4     3.8  3.4   68   57
 K2HgI4     3.4  3.0   63   46
 K2HgI4    15.0  3.0   79   58
 K3IrCl6   15.0  8.3   33   27
 K3IrCl6    8.3  6.4   40   23
 K3IrCl6    6.4  5.2   31   22
 K3IrCl6    5.2  4.4   27   23
 K3IrCl6    4.4  3.8   22   23
 K3IrCl6    3.8  3.4   19   20
 K3IrCl6    3.4  3.0   16   20
 K3IrCl6   15.0  3.0   23   21
```

* **_phasing_MIR_der_shell.d_res_high** (*float*)
The lowest value for the interplanar spacings for the reflection data for this derivative in this shell. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

* **_phasing_MIR_der_shell.d_res_low** (*float*)
The highest value for the interplanar spacings for the reflection data for this derivative in this shell. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

* **_phasing_MIR_der_shell.der_id**
This data item is a pointer to **_phasing_MIR_der.id** in the PHASING_MIR_DER category.

_phasing_MIR_der_shell.fom (*float*)
The mean value of the figure of merit m for reflections in this shell.

$$m = \frac{\int P_\alpha \exp(i\alpha)\,{\rm d}\alpha}{\int P_\alpha\,{\rm d}\alpha},$$

where P_α = the probability that the phase angle α is correct; the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

_phasing_MIR_der_shell.ha_ampl (*float*)
The mean heavy-atom amplitude for reflections for this derivative in this shell.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

_phasing_MIR_der_shell.loc (*float*)
The mean lack-of-closure error loc for reflections for this derivative in this shell.

$$\mathrm{loc} = \sum |F(ph)_{\mathrm{obs}} - F(ph)_{\mathrm{calc}}|,$$

where $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the derivative and $F(ph)_{\mathrm{calc}}$ = the calculated structure-factor amplitude of the derivative; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

_phasing_MIR_der_shell.phase (*float*)
The mean of the phase values for reflections for this derivative in this shell.
[phasing_MIR_der_shell]

_phasing_MIR_der_shell.power (*float*)
The mean phasing power P for reflections for this derivative in this shell.

$$P = \left(\frac{\sum |F(h)^2_{\mathrm{calc}}|}{\sum |F(ph)_{\mathrm{obs}} - F(ph)_{\mathrm{calc}}|^2} \right)^{1/2},$$

where $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of this derivative, $F(ph)_{\mathrm{calc}}$ = the calculated structure-factor amplitude of this derivative and $F(h)_{\mathrm{calc}}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

_phasing_MIR_der_shell.R_cullis (*float*)
Residual factor R_{cullis} for centric reflections for this derivative in this shell.

$$R_{\mathrm{cullis}} = \frac{\sum \left| |F(ph)_{\mathrm{obs}} \pm F(p)_{\mathrm{obs}}| - F(h)_{\mathrm{calc}} \right|}{\sum |F(ph)_{\mathrm{obs}} - F(p)_{\mathrm{obs}}|},$$

where $F(p)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the native, $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the derivative and $F(h)_{\mathrm{calc}}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.

Reference: Cullis, A. F., Muirhead, H., Perutz, M. F., Rossmann, M. G. & North, A. C. T. (1961). *Proc. R. Soc. London Ser. A*, **265**, 15–38.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

_phasing_MIR_der_shell.R_kraut (*float*)
Residual factor R_{kraut} for general reflections for this derivative in this shell.

$$R_{\mathrm{kraut}} = \frac{\sum |F(ph)_{\mathrm{obs}} - F(ph)_{\mathrm{calc}}|}{\sum |F(ph)_{\mathrm{obs}}|},$$

where $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the derivative, $F(ph)_{\mathrm{calc}}$ = the calculated structure-factor amplitude of the derivative and the sum is taken over the specified reflections.

Reference: Kraut, J., Sieker, L. C., High, D. F. & Freer, S. T. (1962). *Proc. Natl Acad. Sci. USA*, **48**, 1417–1424.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_der_shell]**

_phasing_MIR_der_shell.reflns (*int*)
The number of reflections in this shell.
The permitted range is $[0, \infty)$. **[phasing_MIR_der_shell]**

PHASING_MIR_DER_SITE

Data items in the PHASING_MIR_DER_SITE category record details about the heavy-atom sites in an MIR phasing experiment. This list may contain information from a number of different derivatives; **_phasing_MIR_der_site.der_id** indicates to which derivative a given record corresponds. (A derivative in this context does not necessarily equate with a data set; see the definition of the PHASING_MIR_DER category for a discussion of the meaning of 'derivative'.)
Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MIR_der_site.der_id**
_phasing_MIR_der_site.id

*Example 1 – based on a paper by Zanotti et al. [J. Biol. Chem. (1993), **268**, 10728–10738] with occupancies converted from electrons to fractional.*

```
loop_
_phasing_MIR_der_site.der_id
_phasing_MIR_der_site.id
_phasing_MIR_der_site.atom_type_symbol
_phasing_MIR_der_site.occupancy
_phasing_MIR_der_site.fract_x
_phasing_MIR_der_site.fract_y
_phasing_MIR_der_site.fract_z
_phasing_MIR_der_site.B_iso
 KAu(CN)2  1  Au  0.40  0.082  0.266  0.615  33.0
 KAu(CN)2  2  Au  0.03  0.607  0.217  0.816  25.9
 KAu(CN)2  3  Au  0.02  0.263  0.782  0.906  15.7
 K2HgI4    1  Hg  0.63  0.048  0.286  0.636  33.7
 K2HgI4    2  Hg  0.34  0.913  0.768  0.889  36.7
 K2HgI4    3  Hg  0.23  0.974  0.455  0.974  24.2
 K2HgI4    4  Hg  0.28  0.903  0.836  0.859  14.7
 K2HgI4    5  Hg  0.07  0.489  0.200  0.885   6.4
 K2HgI4    6  Hg  0.07  0.162  0.799  0.889  32.9
 K3IrCl6   1  Ir  0.26  0.209  0.739  0.758  40.8
 K3IrCl6   2  Ir  0.05  0.279  0.613  0.752  24.9
```

* **_phasing_MIR_der_site.atom_type_symbol**
This data item is a pointer to **_atom_type.symbol** in the ATOM_TYPE category. The scattering factors referenced *via* this data item should be those used in the refinement of the heavy-atom data; in some cases this is the scattering factor for the single heavy atom, in other cases these are the scattering factors for an atomic cluster.

_phasing_MIR_der_site.B_iso (*float, su*)
Isotropic displacement parameter for this heavy-atom site in this derivative.
Related item: **_phasing_MIR_der_site.B_iso_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.B_iso_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.B_iso**.
Related item: **_phasing_MIR_der_site.B_iso** (associated value).
[phasing_MIR_der_site]

_phasing_MIR_der_site.Cartn_x (*float, su*)
The x coordinate of this heavy-atom position in this derivative specified as orthogonal ångströms. The orthogonal Cartesian axes are related to the cell axes as specified by the description given in **_atom_sites.Cartn_transform_axes**.
Related item: **_phasing_MIR_der_site.Cartn_x_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.Cartn_x_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.Cartn_x**.
Related item: **_phasing_MIR_der_site.Cartn_x** (associated value).
[phasing_MIR_der_site]

_phasing_MIR_der_site.Cartn_y (*float, su*)
The *y* coordinate of this heavy-atom position in this derivative specified as orthogonal ångströms. The orthogonal Cartesian axes are related to the cell axes as specified by the description given in **_atom_sites.Cartn_transform_axes**.
Related item: **_phasing_MIR_der_site.Cartn_y_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.Cartn_y_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.Cartn_y**.
Related item: **_phasing_MIR_der_site.Cartn_y** (associated value).
[phasing_MIR_der_site]

_phasing_MIR_der_site.Cartn_z (*float, su*)
The *z* coordinate of this heavy-atom position in this derivative specified as orthogonal ångströms. The orthogonal Cartesian axes are related to the cell axes as specified by the description given in **_atom_sites.Cartn_transform_axes**.
Related item: **_phasing_MIR_der_site.Cartn_z_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.Cartn_z_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.Cartn_z**.
Related item: **_phasing_MIR_der_site.Cartn_z** (associated value).
[phasing_MIR_der_site]

* **_phasing_MIR_der_site.der_id**
This data item is a pointer to **_phasing_MIR_der.id** in the PHASING_MIR_DER category.

_phasing_MIR_der_site.details (*text*)
A description of special aspects of the derivative site.
Examples: '`binds to His 117`',
'`minor site obtained from difference Fourier`',
'`same as site 2 in the K2HgI4 derivative`'.
[phasing_MIR_der_site]

_phasing_MIR_der_site.fract_x (*float, su*)
The *x* coordinate of this heavy-atom position in this derivative specified as a fraction of **_cell.length_a**.
Related item: **_phasing_MIR_der_site.fract_x_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.fract_x_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.fract_x**.
Related item: **_phasing_MIR_der_site.fract_x** (associated value).
[phasing_MIR_der_site]

_phasing_MIR_der_site.fract_y (*float, su*)
The *y* coordinate of this heavy-atom position in this derivative specified as a fraction of **_cell.length_b**.
Related item: **_phasing_MIR_der_site.fract_y_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.fract_y_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.fract_y**.
Related item: **_phasing_MIR_der_site.fract_y** (associated value).
[phasing_MIR_der_site]

_phasing_MIR_der_site.fract_z (*float, su*)
The *z* coordinate of this heavy-atom position in this derivative specified as a fraction of **_cell.length_c**.
Related item: **_phasing_MIR_der_site.fract_z_esd** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.fract_z_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.fract_z**.
Related item: **_phasing_MIR_der_site.fract_z** (associated value).
[phasing_MIR_der_site]

* **_phasing_MIR_der_site.id** (*code*)
The value of **_phasing_MIR_der_site.id** must uniquely identify each site in each derivative in the PHASING_MIR_DER_SITE list. The atom identifiers need not be unique over all sites in all derivatives; they need only be unique for each site in each derivative. Note that this item need not be a number; it can be any unique identifier.
[phasing_MIR_der_site]

_phasing_MIR_der_site.occupancy (*float*)
The fraction of the atom type present at this heavy-atom site in a given derivative. The sum of the occupancies of all the atom types at this site may not significantly exceed 1.0 unless it is a dummy site.
The permitted range is $[0.0, \infty)$. Where no value is given, the assumed value is '`1.0`'.
[phasing_MIR_der_site]

_phasing_MIR_der_site.occupancy_anom (*float, su*)
_phasing_MIR_der_site.ebi_occupancy_anom (*ebi_extensions 1.0*)
The relative anomalous occupancy of the atom type present at this heavy-atom site in a given derivative. This atom occupancy will probably be on an arbitrary scale.
Related item: **_phasing_MIR_der_site.occupancy_anom_su** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.occupancy_anom_su (*float*)
_phasing_MIR_der_site.ebi_occupancy_anom_esd (*ebi_extensions 1.0*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.occupancy_anom**.
Related item: **_phasing_MIR_der_site.occupancy_anom** (associated value).
[phasing_MIR_der_site]

_phasing_MIR_der_site.occupancy_iso (*float, su*)
_phasing_MIR_der_site.ebi_occupancy_iso (*ebi_extensions 1.0*)
The relative real isotropic occupancy of the atom type present at this heavy-atom site in a given derivative. This atom occupancy will probably be on an arbitrary scale.
Related item: **_phasing_MIR_der_site.occupancy_iso_su** (associated esd).
[phasing_MIR_der_site]

_phasing_MIR_der_site.occupancy_iso_su (*float*)
_phasing_MIR_der_site.ebi_occupancy_iso_esd (*ebi_extensions 1.0*)
The standard uncertainty (estimated standard deviation) of **_phasing_MIR_der_site.occupancy_iso**.
Related item: **_phasing_MIR_der_site.occupancy_iso** (associated value).
[phasing_MIR_der_site]

PHASING_MIR_SHELL

Data items in the PHASING_MIR_SHELL category record statistics for an isomorphous replacement phasing experiment broken down into shells of resolution.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_MIR_shell.d_res_low**
_phasing_MIR_shell.d_res_high

*Example 1 – based on a paper by Zanotti et al. [J. Biol. Chem. (1993), **268**, 10728–10738] with addition of an arbitrary low-resolution limit.*

```
loop_
_phasing_MIR_shell.d_res_low
_phasing_MIR_shell.d_res_high
_phasing_MIR_shell.reflns
_phasing_MIR_shell.FOM
 15.0  8.3   80  0.69
  8.3  6.4  184  0.73
  6.4  5.2  288  0.72
  5.2  4.4  406  0.65
  4.4  3.8  554  0.54
  3.8  3.4  730  0.53
  3.4  3.0  939  0.50
```

* **_phasing_MIR_shell.d_res_high** (*float*)
The lowest value for the interplanar spacings for the reflection data in this shell. This is called the highest resolution. Note that the resolution limits of shells in the items **_phasing_MIR_shell.d_res_high** and **_phasing_MIR_shell.d_res_low** are independent of the resolution limits of shells in the items **_reflns_shell.d_res_high** and **_reflns_shell.d_res_low**.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

* **_phasing_MIR_shell.d_res_low** (*float*)
The highest value for the interplanar spacings for the reflection data in this shell. This is called the lowest resolution. Note that the resolution limits of shells in the items **_phasing_MIR_shell.d_res_high** and **_phasing_MIR_shell.d_res_low** are independent of the resolution limits of shells in the items **_reflns_shell.d_res_high** and **_reflns_shell.d_res_low**.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.FOM (*float*)
The mean value of the figure of merit m for reflections in this shell.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha \, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct and the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.FOM_acentric (*float*)
_phasing_MIR_shell.ebi_fom_acentric (*ebi_extensions 1.0*)
The mean value of the figure of merit m for acentric reflections in this shell.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha \, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct and the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.FOM_centric (*float*)
_phasing_MIR_shell.ebi_fom_centric (*ebi_extensions 1.0*)
The mean value of the figure of merit m for centric reflections in this shell.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha \, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct and the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.loc (*float*)
The mean lack-of-closure error loc for reflections in this shell.

$$\mathrm{loc} = \sum |F(ph)_{\mathrm{obs}} - F(ph)_{\mathrm{calc}}|,$$

where $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the derivative and $F(ph)_{\mathrm{calc}}$ = the calculated structure-factor amplitude of the derivative; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.mean_phase (*float*)
The mean of the phase values for all reflections in this shell.
[phasing_MIR_shell]

_phasing_MIR_shell.power (*float*)
The mean phasing power P for reflections in this shell.

$$P = \left(\frac{\sum |F(h)^2_{\mathrm{calc}}|}{\sum |F(ph)_{\mathrm{obs}} - F(ph)_{\mathrm{calc}}|^2} \right)^{1/2},$$

where $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of this derivative, $F(ph)_{\mathrm{calc}}$ = the calculated structure-factor amplitude of this derivative and $F(h)_{\mathrm{calc}}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.R_cullis (*float*)
Residual factor R_{cullis} for centric reflections in this shell.

$$R_{\mathrm{cullis}} = \frac{\sum \Big| |F(ph)_{\mathrm{obs}} \pm F(p)_{\mathrm{obs}}| - F(h)_{\mathrm{calc}} \Big|}{\sum |F(ph)_{\mathrm{obs}} - F(p)_{\mathrm{obs}}|},$$

where $F(p)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the native, $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the derivative and $F(h)_{\mathrm{calc}}$ = the calculated structure-factor amplitude from the heavy-atom model; the sum is taken over the specified reflections.

Reference: Cullis, A. F., Muirhead, H., Perutz, M. F., Rossmann, M. G. & North, A. C. T. (1961). *Proc. R. Soc. London Ser. A*, **265**, 15–38.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.R_kraut (*float*)
Residual factor R_{kraut} for general reflections in this shell.

$$R_{\mathrm{kraut}} = \frac{\sum |F(ph)_{\mathrm{obs}} - F(ph)_{\mathrm{calc}}|}{\sum |F(ph)_{\mathrm{obs}}|},$$

where $F(ph)_{\mathrm{obs}}$ = the observed structure-factor amplitude of the derivative and $F(ph)_{\mathrm{calc}}$ = the calculated structure-factor amplitude of the derivative; the sum is taken over the specified reflections.

Reference: Kraut, J., Sieker, L. C., High, D. F. & Freer, S. T. (1962). *Proc. Natl Acad. Sci. USA*, **48**, 1417–1424.
The permitted range is $[0.0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.reflns (*int*)
The number of reflections in this shell.
The permitted range is $[0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.reflns_acentric (*int*)
_phasing_MIR_shell.ebi_reflns_acentric (*ebi_extensions 1.0*)
The number of acentric reflections in this shell.
The permitted range is $[0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.reflns_anomalous (*int*)
The number of anomalous reflections in this shell.
The permitted range is $[0, \infty)$. **[phasing_MIR_shell]**

_phasing_MIR_shell.reflns_centric (*int*)
_phasing_MIR_shell.ebi_reflns_centric (*ebi_extensions 1.0*)
The number of centric reflections in this shell.
The permitted range is $[0, \infty)$. **[phasing_MIR_shell]**

PHASING_SET

Data items in the PHASING_SET category record details about the data sets used in a phasing experiment. A given data set may be used in a number of different ways; for instance, a single data set could be used both as an isomorphous derivative and as a component of a multiple-wavelength calculation. This category establishes identifiers for each data set and permits the archiving of a subset of experimental information for each data set (cell constants, wavelength, temperature *etc.*). This and related categories of data items are provided so that derivative intensity and phase information can be stored in the same data block as the information for the refined structure. If all the possible experimental information for each data set (raw data sets, crystal growth conditions *etc.*) is to be archived, these data items should be recorded in a separate data block.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_set.id**

Example 1 – based on laboratory records for an Hg/Pt derivative of protein NS1.

```
_phasing_set.id                    'NS1-96'
_phasing_set.cell_angle_alpha      90.0
_phasing_set.cell_angle_beta       90.0
_phasing_set.cell_angle_gamma      90.0
_phasing_set.cell_length_a         38.63
_phasing_set.cell_length_b         38.63
_phasing_set.cell_length_c         82.88
_phasing_set.radiation_wavelength  1.5145
_phasing_set.detector_type         'image plate'
_phasing_set.detector_specific     'RXII'
```

_phasing_set.cell_angle_alpha (*float*)
Unit-cell angle α for this data set in degrees.
The permitted range is [0.0, 180.0]. Where no value is given, the assumed value is '90.0'. **[phasing_set]**

_phasing_set.cell_angle_beta (*float*)
Unit-cell angle β for this data set in degrees.
The permitted range is [0.0, 180.0]. Where no value is given, the assumed value is '90.0'. **[phasing_set]**

_phasing_set.cell_angle_gamma (*float*)
Unit-cell angle γ for this data set in degrees.
The permitted range is [0.0, 180.0]. Where no value is given, the assumed value is '90.0'. **[phasing_set]**

_phasing_set.cell_length_a (*float*)
Unit-cell length *a* for this data set in ångströms.
The permitted range is $[0.0, \infty)$. **[phasing_set]**

_phasing_set.cell_length_b (*float*)
Unit-cell length *b* for this data set in ångströms.
The permitted range is $[0.0, \infty)$. **[phasing_set]**

_phasing_set.cell_length_c (*float*)
Unit-cell length *c* for this data set in ångströms.
The permitted range is $[0.0, \infty)$. **[phasing_set]**

_phasing_set.detector_specific (*text*)
The particular radiation detector. In general, this will be a manufacturer, description, model number or some combination of these.
Examples: 'Siemens model x', 'Kodak XG', 'MAR Research model y'. **[phasing_set]**

_phasing_set.detector_type (*text*)
The general class of the radiation detector.
Examples: 'multiwire', 'imaging plate', 'CCD', 'film'. **[phasing_set]**

* **_phasing_set.id** (*line*)
The value of **_phasing_set.id** must uniquely identify a record in the PHASING_SET list. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_phasing_set_refln.set_id,
_phasing_MAD_set.set_id,
_phasing_MIR_der.der_set_id,
_phasing_MIR_der.native_set_id,
_phasing_MIR_der_refln.set_id.
Examples: 'KAu(CN)2', 'K2HgI4'. **[phasing_set]**

_phasing_set.radiation_source_specific (*text*)
The particular source of radiation. In general, this will be a manufacturer, description, or model number (or some combination of these) for laboratory sources and an institution name and beamline name for synchrotron sources.
Examples: 'Rigaku RU200', 'Philips fine focus Mo', 'NSLS beamline X8C'. **[phasing_set]**

_phasing_set.radiation_wavelength (*float*)
The mean wavelength of the radiation used to measure this data set.
The permitted range is $[0.0, \infty)$. **[phasing_set]**

_phasing_set.temp (*float*)
The temperature in kelvins at which the data set was measured.
The permitted range is $[0.0, \infty)$. **[phasing_set]**

PHASING_SET_REFLN

Data items in the PHASING_SET_REFLN category record the values of the measured structure factors used in a phasing experiment. This list may contain information from a number of different data sets; **_phasing_set_refln.set_id** indicates the data set to which a given record corresponds.

Category group(s): **inclusive_group**
phasing_group
Category key(s): **_phasing_set_refln.index_h**
_phasing_set_refln.index_k
_phasing_set_refln.index_l
_phasing_set_refln.set_id

Example 1 – based on laboratory records for the 15,15,32 reflection of an Hg/Pt derivative of protein NS1.

```
_phasing_set_refln.set_id           'NS1-96'
_phasing_set_refln.index_h           15
_phasing_set_refln.index_k           15
_phasing_set_refln.index_l           32
_phasing_set_refln.F_meas_au        181.79
_phasing_set_refln.F_meas_sigma_au    3.72
```

_phasing_set_refln.F_meas (*float, su*)
The measured value of the structure factor for this reflection in this data set in electrons.
Related items: **_phasing_set_refln.F_meas_sigma** (associated esd), **_phasing_set_refln.F_meas_au** (conversion arbitrary).
[phasing_set_refln]

_phasing_set_refln.F_meas_au (*float, su*)
The measured value of the structure factor for this reflection in this data set in arbitrary units.
Related items: **_phasing_set_refln.F_meas_sigma_au** (associated esd), **_phasing_set_refln.F_meas** (conversion arbitrary). **[phasing_set_refln]**

_phasing_set_refln.F_meas_sigma (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_set_refln.F_meas** in electrons.
Related items: **_phasing_set_refln.F_meas** (associated value), **_phasing_set_refln.F_meas_sigma_au** (conversion arbitrary).
[phasing_set_refln]

_phasing_set_refln.F_meas_sigma_au (*float*)
The standard uncertainty (estimated standard deviation) of **_phasing_set_refln.F_meas_au** in arbitrary units.
Related items: **_phasing_set_refln.F_meas_au** (associated value), **_phasing_set_refln.F_meas_sigma** (conversion arbitrary).
[phasing_set_refln]

***_phasing_set_refln.index_h** (*int*)
Miller index *h* of this reflection in this data set.
[phasing_set_refln]

***_phasing_set_refln.index_k** (*int*)
Miller index *k* of this reflection in this data set.
[phasing_set_refln]

***_phasing_set_refln.index_l** (*int*)
Miller index *l* of this reflection in this data set.
[phasing_set_refln]

***_phasing_set_refln.set_id**
This data item is a pointer to **_phasing_set.id** in the PHASING_SET category.

PUBL

Data items in the PUBL category are used when submitting a manuscript for publication.

Category group(s): **inclusive_group**
iucr_group
Category key(s): **_publ.entry_id**

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_publ.section_title
;  trans-3-Benzoyl-2-(tert-butyl)-4-(iso-butyl)-
   1,3-oxazolidin-5-one
;

_publ.section_abstract
;  The oxazolidinone ring is a shallow envelope
   conformation with the tert-butyl and iso-butyl groups
   occupying trans-positions with respect to the ring.  The
   angles at the N atom sum to 356.2\%, indicating a very
   small degree of pyramidalization at this atom.  This is
   consistent with electron delocalization between the N
   atom and the carbonyl centre [N-C=O = 1.374(3)\%A].
;
```

*Example 2 – based on $C_{31}H_{48}N_4O_4$, reported by Coleman, Patrick, Andersen & Rettig [Acta Cryst. (1996), C**52**, 1525–1527].*

```
_publ.section_title
;  Hemiasterlin methyl ester
;

_publ.section_title_footnote
;  IUPAC name: methyl 2,5-dimethyl-4-2-[3-methyl-
   2-methylamino-3-(N-methylbenzo[b]pyrrol-
   3-yl)butanamido]-3,3-dimethyl-N-methyl-
   butanamido-2-hexenoate.
;
```

_publ.contact_author (*text*)
_publ_contact_author (*cif_core.dic 2.0.1*)
The name and address of the author submitting the manuscript and data block. This is the person contacted by the journal editorial staff. It is preferable to use the separate data items **_publ.contact_author_name** and **_publ.contact_author_address**.

Example:
```
; Professor George Ferguson
  Department of Chemistry and Biochemistry
  University of Guelph
  Ontario
  Canada
  N1G 2W1
;
```
[publ]

_publ.contact_author_address (*text*)
_publ_contact_author_address (*cif_core.dic 2.0.1*)
The address of the author submitting the manuscript and data block. This is the person contacted by the journal editorial staff.

Example:
```
; Department of Chemistry and Biochemistry
  University of Guelph
  Ontario
  Canada
  N1G 2W1
;
```
[publ]

_publ.contact_author_email (*line*)
_publ_contact_author_email (*cif_core.dic 2.0.1*)
E-mail address in a form recognizable to international networks. The format of e-mail addresses is given in Section 3.4, *Address Specification*, of *Internet Message Format*, RFC 2822, P. Resnick (Editor), Network Standards Group, April 2001.
Examples: 'name@host.domain.country', 'uur5@banjo.bitnet'. **[publ]**

_publ.contact_author_fax (*line*)
_publ_contact_author_fax (*cif_core.dic 2.0.1*)
Facsimile telephone number of the author submitting the manuscript and data block. The recommended style starts with the international dialing prefix, followed by the area code in parentheses, followed by the local number with no spaces. The earlier convention of including the international dialing prefix in parentheses is no longer recommended.
Examples: '12(34)9477330', '12()349477330'. **[publ]**

_publ.contact_author_name (*text*)
_publ_contact_author_name (*cif_core.dic 2.0.1*)
The name of the author submitting the manuscript and data block. This is the person contacted by the journal editorial staff.
Example:
```
; Professor George Ferguson
;
```
[publ]

_publ.contact_author_phone (*line*)
_publ_contact_author_phone (*cif_core.dic 2.0.1*)
Telephone number of the author submitting the manuscript and data block. The recommended style starts with the international dialing prefix, followed by the area code in parentheses, followed by the local number and any extension number prefixed by 'x', with no spaces. The earlier convention of including the international dialing prefix in parentheses is no longer recommended.
Examples: '12(34)9477330', '12()349477330', '12(34)9477330x5543'.
[publ]

_publ.contact_letter (*text*)
_publ_contact_letter (*cif_core.dic 2.0.1*)
A letter submitted to the journal editor by the contact author.
[publ]

***_publ.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_publ.manuscript_creation (*text*)
_publ_manuscript_creation (*cif_core.dic 2.0.1*)
A description of the word-processor package and computer used to create the word-processed manuscript stored as **_publ.manuscript_processed**.
Example: 'Tex file created by FrameMaker on a Sun 3/280'. **[publ]**

_publ.manuscript_processed (*text*)
_publ_manuscript_processed (*cif_core.dic 2.0.1*)
The full manuscript of a paper (excluding possibly the figures and the tables) output in ASCII characters from a word processor. Information about the generation of this data item must be specified in the data item **_publ.manuscript_creation**.
[publ]

_publ.manuscript_text (*text*)
_publ_manuscript_text (*cif_core.dic 2.0.1*)
The full manuscript of a paper (excluding figures and possibly the tables) output as standard ASCII text.
[publ]

_publ.requested_category (*line*)
_publ_requested_category (*cif_core.dic 2.0.1*)
The category of paper submitted. For submission to *Acta Crystallographica Section C* or *Acta Crystallographica Section E*, *only* the codes indicated for use with these journals should be used.
The data value must be one of the following:

FA	Full article
FI	Full submission – inorganic (*Acta C*)
FO	Full submission – organic (*Acta C*)
FM	Full submission – metal-organic (*Acta C*)
CI	CIF-access paper – inorganic (*Acta C*) (no longer in use)
CO	CIF-access paper – organic (*Acta C*) (no longer in use)
CM	CIF-access paper – metal-organic (*Acta C*) (no longer in use)
EI	Electronic submission – inorganic (*Acta E*)
EO	Electronic submission – organic (*Acta E*)
EM	Electronic submission – metal-organic (*Acta E*)
AD	Addenda and Errata (*Acta C*, *Acta E*)
SC	Short communication

Where no value is given, the assumed value is 'FA'. **[publ]**

_publ.requested_coeditor_name (*line*)
_publ_requested_coeditor_name (*cif_core.dic 2.0.1*)
The name of the co-editor whom the authors would like to handle the submitted manuscript.
[publ]

_publ.requested_journal (*line*)
_publ_requested_journal (*cif_core.dic 2.0.1*)
The name of the journal to which the manuscript is being submitted.
[publ]

_publ.section_abstract (*text*)
_publ_section_abstract (*cif_core.dic 2.0.1*)
The abstract section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_acknowledgements (*text*)
_publ_section_acknowledgements (*cif_core.dic 2.0.1*)
The acknowledgements section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_comment (*text*)
_publ_section_comment (*cif_core.dic 2.0.1*)
The comment section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_discussion (*text*)
_publ_section_discussion (*cif_core.dic 2.0.1*)
The discussion section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_experimental (*text*)
_publ_section_experimental (*cif_core.dic 2.0.1*)
The experimental section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**. The **_publ.section_exptl_prep**, **_publ.section_exptl_solution** and **_publ.section_exptl_refinement** items are preferred for separating the chemical preparation, structure solution and refinement aspects of the description of the experiment.
[publ]

_publ.section_exptl_prep (*text*)
_publ_section_exptl_prep (*cif_core.dic 2.0.1*)
The experimental preparation section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_exptl_refinement (*text*)
_publ_section_exptl_refinement (*cif_core.dic 2.0.1*)
The experimental refinement section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_exptl_solution (*text*)
_publ_section_exptl_solution (*cif_core.dic 2.0.1*)
The experimental solution section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_figure_captions (*text*)
_publ_section_figure_captions (*cif_core.dic 2.0.1*)
The figure captions section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_introduction (*text*)
_publ_section_introduction (*cif_core.dic 2.0.1*)
The introduction section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_references (*text*)
_publ_section_references (*cif_core.dic 2.0.1*)
The references section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_synopsis (*text*)
_publ_section_synopsis (*cif_core.dic 2.0.1*)
The synopsis section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_table_legends (*text*)
_publ_section_table_legends (*cif_core.dic 2.0.1*)
The table legends section of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_title (*text*)
_publ_section_title (*cif_core.dic 2.0.1*)
The title of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

_publ.section_title_footnote (*text*)
_publ_section_title_footnote (*cif_core.dic 2.0.1*)
The footnote to the title of a manuscript if the manuscript is submitted in parts. As an alternative see **_publ.manuscript_text** and **_publ.manuscript_processed**.
[publ]

PUBL_AUTHOR

Data items in the PUBL_AUTHOR category record details of the authors of a manuscript submitted for publication.
Category group(s): **inclusive_group**
iucr_group
Category key(s): **_publ_author.name**

*Example 1 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
loop_
_publ_author.name
_publ_author.address
     'Willis, Anthony C.'
;     Research School of Chemistry
      Australian National University
      GPO Box 4
      Canberra, A.C.T.
      Australia    2601
;
```

_publ_author.address (*text*)
_publ_author_address (*cif_core.dic 2.0.1*)
The address of a publication author. If there is more than one author this is looped with **_publ_author.name**.
Example:
```
; Department
  Institute
  Street
  City and postcode
  COUNTRY
;
```
[publ_author]

_publ_author.email (*text*)
_publ_author_email (*cif_core.dic 2.3.1*)
The e-mail address of a publication author. If there is more than one author, this will be looped with **_publ_author.name**. The format of e-mail addresses is given in Section 3.4, *Address Specification*, of *Internet Message Format*, RFC 2822, P. Resnick (Editor), Network Standards Group, April 2001.
Examples: '`name@host.domain.country`', '`bm@iucr.org`'. **[publ_author]**

_publ_author.footnote (*line*)
_publ_author_footnote (*cif_core.dic 2.0.1*)
A footnote accompanying an author's name in the list of authors of a paper. Typically indicates sabbatical address, additional affiliations or date of decease.
Examples: '`On leave from U. Western Australia`',
'`Also at Department of Biophysics`'. **[publ_author]**

_publ_author.id_iucr (*code*)

_publ_author_id_iucr (*cif_core.dic 2.3*)

Identifier in the IUCr contact database of a publication author. This identifier may be available from the *World Directory of Crystallographers* (http://wdc.iucr.org).

Example: '2985'. **[publ_author]**

*_publ_author.name (*line*)

_publ_author_name (*cif_core.dic 2.0.1*)

The name of a publication author. If there are multiple authors this will be looped with **_publ_author.address**. The family name(s), followed by a comma and including any dynastic components, precedes the first names or initials.

Examples: 'Bleary, Percival R.', 'O'Neil, F.K.', 'Van den Bossche, G.', 'Yang, D.-L.', 'Simonov, Yu.A.'. **[publ_author]**

PUBL_BODY

Data items in the PUBL_BODY category permit the labelling of different text sections within the body of a paper. Note that these should not be used in a paper which has a standard format with sections tagged by specific data names (such as in *Acta Crystallographica Section C*). Typically, each journal will supply a list of the specific items it requires in its *Notes for Authors*.

Category group(s): **inclusive_group**
iucr_group
Category key(s): **_publ_body.element**
_publ_body.label

*Example 1 – based on a paper by R. Restori & D. Schwarzenbach [Acta Cryst. (1996), A**52**, 369–378].*

```
loop_
_publ_body.element
_publ_body.label
_publ_body.title
_publ_body.format
_publ_body.contents

     section   1         Introduction                    cif
; X-ray diffraction from a crystalline material provides
  information on the thermally and spatially averaged
  electron density in the crystal...
;
     section   2         Theory                         tex
; In the rigid-atom approximation, the dynamic electron
  density of an atom is described by the convolution
  product of the static atomic density and a probability
  density function,
  $\rho_dyn(\bf r) = \rho_stat(\bf r) * P(\bf r). \eqno(1)$
;
```

*Example 2 – based on a paper by R. J. Papoular, Y. Vekhter & P. Coppens [Acta Cryst. (1996), A**52**, 397–407].*

```
loop_
_publ_body.element
_publ_body.label
_publ_body.title
_publ_body.contents

     section     3
; The two-channel method for retrieval of the deformation
  electron density
;
     .
     subsection  3.1  'The two-channel entropy S[\D\r(r)]'
; As the wide dynamic range involved in the total electron
  density...
;
     subsection  3.2
'Uniform vs informative prior model densities'        .
     subsubsection  3.2.1  'Use of uniform models'
; Straightforward algebra leads to expressions analogous
  to...
;
```

_publ_body.contents (*text*)

_publ_body_contents (*cif_core.dic 2.0.1*)

A text section of a paper.

[publ_body]

_publ_body.element (*code*)

_publ_body_element (*cif_core.dic 2.0.1*)

The functional role of the associated text section.

The data value must be one of the following:

- section
- subsection
- subsubsection
- appendix
- footnote

[publ_body]

_publ_body.format (*code*)

_publ_body_format (*cif_core.dic 2.0.1*)

Code indicating the appropriate typesetting conventions for accented characters and special symbols in the text section.

The data value must be one of the following:

ascii	no coding for special symbols
cif	CIF convention
latex	LaTeX
sgml	SGML (ISO 8879)
tex	TeX
troff	troff or nroff

[publ_body]

_publ_body.label (*code*)

_publ_body_label (*cif_core.dic 2.0.1*)

Code identifying the section of text.

Examples: '1', '1.1', '2.1.3'. **[publ_body]**

_publ_body.title (*text*)

_publ_body_title (*cif_core.dic 2.0.1*)

Title of the associated section of text.

[publ_body]

PUBL_MANUSCRIPT_INCL

Data items in the PUBL_MANUSCRIPT_INCL category allow the authors of a manuscript submitted for publication to list data names that should be added to the standard request list used by the journal printing software.

Category group(s): **inclusive_group**
iucr_group
Category key(s): **_publ_manuscript_incl.entry_id**

Example 1 – hypothetical example.

```
_publ_manuscript_incl.entry_id   'EXAMHYPO'
loop_
_publ_manuscript_incl.extra_item
_publ_manuscript_incl.extra_info
_publ_manuscript_incl.extra_defn
'_atom_site.symmetry_multiplicity'
     'to emphasise special sites'  yes
'_chemical.compound_source'
     'rare material, unusual source'   yes
'_reflns.d_resolution_high'
     'limited data are a problem here'  yes
'_crystal.magnetic_permeability'
     'unusual value for this material'  no
```

*_publ_manuscript_incl.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

_publ_manuscript_incl.extra_defn (*line*)
_publ_manuscript_incl_extra_defn (*cif_core.dic 2.0.1*)

Flags whether the corresponding data item marked for inclusion in a journal request list is a standard CIF definition or not.

The data value must be one of the following:

no	not a standard CIF data name
n	abbreviation for 'no'
yes	a standard CIF data name
y	abbreviation for 'yes'

[publ_manuscript_incl]

_publ_manuscript_incl.extra_info (*text*)
_publ_manuscript_incl_extra_info (*cif_core.dic 2.0.1*)

A short note indicating the reason why the author wishes the corresponding data item marked for inclusion in the journal request list to be published.

Examples: 'to emphasise very special sites',
'rare material from unusual source',
'the limited data is a problem here',
'a new data quantity needed here'. **[publ_manuscript_incl]**

_publ_manuscript_incl.extra_item (*line*)
_publ_manuscript_incl_extra_item (*cif_core.dic 2.0.1*)

Specifies the inclusion of specific data into a manuscript which are not normally requested by the journal. The values of this item are the extra data names (which *must* be enclosed in single quotes) that will be added to the journal request list.

Examples: '_atom_site.symmetry_multiplicity',
'_chemical.compound_source', '_reflns.d_resolution_high',
'_crystal.magnetic_permeability'. **[publ_manuscript_incl]**

REFINE

Data items in the REFINE category record details about the structure-refinement parameters.

Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_refine.entry_id                          '5HVP'
_refine.ls_number_reflns_obs              12901
_refine.ls_number_restraints              6609
_refine.ls_number_parameters              7032
_refine.ls_R_Factor_obs                   0.176
_refine.ls_weighting_scheme               calc
_refine.ls_weighting_details
; Sigdel model of Konnert-Hendrickson:
  Sigdel: Afsig +  Bfsig*(sin(theta)/lambda-1/6)
  Afsig = 22.0, Bfsig = -150.0 at beginning of refinement
  Afsig = 15.5, Bfsig =  -50.0 at end of refinement
;
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), **C47**, 2276–2277].*

```
_refine.details              sfls:_F_calc_weight_full_matrix

_refine.ls_structure_factor_coef  F
_refine.ls_matrix_type            full
_refine.ls_weighting_scheme       calc
_refine.ls_weighting_details     'w=1/(\s^2^(F)+0.0004F^2^)'
_refine.ls_hydrogen_treatment    'refxyz except H332B noref'
_refine.ls_extinction_method      Zachariasen
_refine.ls_extinction_coef        3514
_refine.ls_extinction_expression
; Larson, A. C. (1970). "Crystallographic Computing", edited
  by F. R. Ahmed. Eq. (22) p. 292. Copenhagen: Munksgaard.

_refine.ls_abs_structure_details
; The absolute configuration was assigned to agree with the
  known chirality at C3 arising from its precursor l-leucine.
;
_refine.ls_abs_structure_Flack      0
_refine.ls_number_reflns_obs     1408
_refine.ls_number_parameters      272
_refine.ls_number_restraints        0
_refine.ls_number_constraints       0
_refine.ls_R_factor_all              .038
_refine.ls_R_factor_obs              .034
_refine.ls_wR_factor_all             .044
_refine.ls_wR_factor_obs             .042
_refine.ls_goodness_of_fit_all      1.462
_refine.ls_goodness_of_fit_obs      1.515
_refine.ls_shift_over_esd_max        .535
_refine.ls_shift_over_esd_mean       .044
_refine.diff_density_min            -.108
_refine.diff_density_max             .131
```

_refine.aniso_B[1][1] (*float*)

The [1][1] element of the matrix that defines the overall anisotropic displacement model if one was refined for this structure.

[refine]

_refine.aniso_B[1][2] (*float*)

The [1][2] element of the matrix that defines the overall anisotropic displacement model if one was refined for this structure.

[refine]

_refine.aniso_B[1][3] (*float*)

The [1][3] element of the matrix that defines the overall anisotropic displacement model if one was refined for this structure.

[refine]

_refine.aniso_B[2][2] (*float*)

The [2][2] element of the matrix that defines the overall anisotropic displacement model if one was refined for this structure.

[refine]

_refine.aniso_B[2][3] (*float*)

The [2][3] element of the matrix that defines the overall anisotropic displacement model if one was refined for this structure.

[refine]

_refine.aniso_B[3][3] (*float*)

The [3][3] element of the matrix that defines the overall anisotropic displacement model if one was refined for this structure.

[refine]

_refine.B_iso_max (*float*)

The maximum isotropic displacement parameter (B value) found in the coordinate set.

[refine]

_refine.B_iso_mean (*float*)
The mean isotropic displacement parameter (B value) for the coordinate set.
[refine]

_refine.B_iso_min (*float*)
The minimum isotropic displacement parameter (B value) found in the coordinate set.
[refine]

_refine.correlation_coeff_Fo_to_Fc (*float*)
_refine.ebi_Correlation_coeff_Fo_to_Fc (*ebi_extensions 1.0*)
The correlation coefficient between the observed and calculated structure factors for reflections included in the refinement. The correlation coefficient is scale-independent and gives an idea of the quality of the refined model.

$$R_{\rm corr} = \frac{\sum_i (F_{oi}F_{ci} - \langle F_o\rangle\langle F_c\rangle)}{\sqrt{\sum_i (F_{oi})^2 - \langle F_o\rangle^2}\sqrt{\sum_i (F_{ci})^2 - \langle F_c\rangle^2}},$$

where F_o = observed structure factors, F_c = calculated structure factors, $\langle\rangle$ denotes average value and the summation is over reflections included in the refinement.
[refine]

_refine.correlation_coeff_Fo_to_Fc_free (*float*)
_refine.ebi_Correlation_coeff_Fo_to_Fc_free (*ebi_extensions 1.0*)
The correlation coefficient between the observed and calculated structure factors for reflections not included in the refinement (free reflections). The correlation coefficient is scale-independent and gives an idea of the quality of the refined model.

$$R_{\rm corr} = \frac{\sum_i (F_{oi}F_{ci} - \langle F_o\rangle\langle F_c\rangle)}{\sqrt{\sum_i (F_{oi})^2 - \langle F_o\rangle^2}\sqrt{\sum_i (F_{ci})^2 - \langle F_c\rangle^2}},$$

where F_o = observed structure factors, F_c = calculated structure factors, $\langle\rangle$ denotes average value and the summation is over reflections not included in the refinement (free reflections).
[refine]

_refine.details (*text*)
_refine_special_details (*cif_core.dic 2.0.1*)
Description of special aspects of the refinement process.
[refine]

_refine.diff_density_max (*float, su*)
_refine_diff_density_max (*cif_core.dic 2.0.1*)
The maximum value of the electron density in the final difference Fourier map.
Related item: **_refine.diff_density_max_esd** (associated esd). **[refine]**

_refine.diff_density_max_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.diff_density_max**.
Related item: **_refine.diff_density_max** (associated value). **[refine]**

_refine.diff_density_min (*float, su*)
_refine_diff_density_min (*cif_core.dic 2.0.1*)
The minimum value of the electron density in the final difference Fourier map.
Related item: **_refine.diff_density_min_esd** (associated esd). **[refine]**

_refine.diff_density_min_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.diff_density_min**.
Related item: **_refine.diff_density_min** (associated value). **[refine]**

_refine.diff_density_rms (*float, su*)
_refine_diff_density_rms (*cif_core.dic 2.0.1*)
The root-mean-square-deviation of the electron density in the final difference Fourier map. This value is measured with respect to the arithmetic mean density and is derived from summations over each grid point in the asymmetric unit of the cell. This quantity is useful for assessing the significance of the values of **_refine.diff_density_min** and **_refine.diff_density_max**, and also for defining suitable contour levels.
Related item: **_refine.diff_density_rms_esd** (associated esd). **[refine]**

_refine.diff_density_rms_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.diff_density_rms**.
Related item: **_refine.diff_density_rms** (associated value). **[refine]**

***_refine.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_refine.ls_abs_structure_details (*text*)
_refine_ls_abs_structure_details (*cif_core.dic 2.0.1*)
The nature of the absolute structure and how it was determined. For example, this may describe the Friedel pairs used.
[refine]

_refine.ls_abs_structure_Flack (*float, su*)
_refine_ls_abs_structure_Flack (*cif_core.dic 2.0.1*)
The measure of absolute structure (enantiomorph or polarity) as defined by Flack (1983). For centrosymmetric structures the only permitted value, if the data name is present, is 'inapplicable', represented by '.' . For noncentrosymmetric structures the value must lie in the 99.97% Gaussian confidence interval $-3u \le x \le 1 + 3u$ and a standard uncertainty (estimated standard deviation) u must be supplied. The item range of $[0.0, 1.0]$ is correctly interpreted as meaning $(0.0 - 3u) \le x \le (1.0 + 3u)$.

Reference: Flack, H. D. (1983). *Acta Cryst.* A**39**, 876–881.

The permitted range is [0.0, 1.0].
Related item: **_refine.ls_abs_structure_Flack_esd** (associated esd).
[refine]

_refine.ls_abs_structure_Flack_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.ls_abs_structure_Flack**.
Related item: **_refine.ls_abs_structure_Flack** (associated value). **[refine]**

_refine.ls_abs_structure_Rogers (*float, su*)
_refine_ls_abs_structure_Rogers (*cif_core.dic 2.0.1*)
The measure of absolute structure (enantiomorph or polarity) as defined by Rogers. The value must lie in the 99.97% Gaussian confidence interval $-1 - 3u \le \eta \le 1 + 3u$ and a standard uncertainty (estimated standard deviation) u must be supplied. The item range of $[-1.0, 1.0]$ is correctly interpreted as meaning $(-1.0 - 3u) \le \eta \le (1.0 + 3u)$.

Reference: Rogers, D. (1981). *Acta Cryst.* A**37**, 734–741.

The permitted range is [−1.0, 1.0].
Related item: **_refine.ls_abs_structure_Rogers_esd** (associated esd).
[refine]

_refine.ls_abs_structure_Rogers_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.ls_abs_structure_Rogers**.
Related item: **_refine.ls_abs_structure_Rogers** (associated value). **[refine]**

***_refine.ls_d_res_high** (*float*)
_refine_ls_d_res_high (*cif_core.dic 2.0.1*)
The smallest value for the interplanar spacings for the reflection data used in the refinement in ångströms. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[refine]**

***_refine.ls_d_res_low** (*float*)
_refine_ls_d_res_low (*cif_core.dic 2.0.1*)
The largest value for the interplanar spacings for the reflection data used in the refinement in ångströms. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_extinction_coef (*float, su*)
_refine_ls_extinction_coef (*cif_core.dic 2.0.1*)
The extinction coefficient used to calculate the correction factor applied to the structure-factor data. The nature of the extinction coefficient is given in the definitions of **_refine.ls_extinction_expression** and **_refine.ls_extinction_method**. For the 'Zachariasen' method it is the r^* value; for the 'Becker–Coppens type 1 isotropic' method it is the 'g' value, and for 'Becker–Coppens type 2 isotropic' corrections it is the 'ρ' value. Note that the magnitude of these values is usually of the order of 10 000.

References: Becker, P. J. & Coppens, P. (1974). *Acta Cryst.* A**30**, 129–147, 148–153. Zachariasen, W. H. (1967). *Acta Cryst.* **23**, 558–564. Larson, A. C. (1967). *Acta Cryst.* **23**, 664–665.
Related item: **_refine.ls_extinction_coef_esd** (associated esd) .
Example: '3472' (Zachariasen coefficient r* = 0.347 E04). **[refine]**

_refine.ls_extinction_coef_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.ls_extinction_coef**.
Related item: **_refine.ls_extinction_coef** (associated value). **[refine]**

_refine.ls_extinction_expression (*text*)
_refine_ls_extinction_expression (*cif_core.dic 2.0.1*)
A description of or reference to the extinction-correction equation used to apply the data item **_refine.ls_extinction_coef**. This information must be sufficient to reproduce the extinction-correction factors applied to the structure factors.
Example:
```
; Larson, A. C. (1970). "Crystallographic Computing", edited
by  F. R. Ahmed. Eq. (22), p.292. Copenhagen: Munksgaard.
;
```
[refine]

_refine.ls_extinction_method (*text*)
_refine_ls_extinction_method (*cif_core.dic 2.0.1*)
A description of the extinction-correction method applied. This description should include information about the correction method, either 'Becker–Coppens' or 'Zachariasen'. The latter is sometimes referred to as the 'Larson' method even though it employs Zachariasen's formula. The Becker–Coppens procedure is referred to as 'type 1' when correcting secondary extinction dominated by the mosaic spread; as 'type 2' when secondary extinction is dominated by particle size and includes a primary extinction component; and as 'mixed' when there is a mixture of types 1 and 2. For the Becker–Coppens method, it is also necessary to set the mosaic distribution as either 'Gaussian' or 'Lorentzian' and the nature of the extinction as 'isotropic' or 'anisotropic'. Note that if either the 'mixed' or 'anisotropic' corrections are applied the multiple coefficients cannot be contained in ***_extinction_coef** and must be listed in **_refine.details**.

References: Becker, P. J. & Coppens, P. (1974). *Acta Cryst.* A**30**, 129–147, 148–153. Zachariasen, W. H. (1967). *Acta Cryst.* **23**, 558–564. Larson, A. C. (1967). *Acta Cryst.* **23**, 664–665.
Example: 'B-C type 2 Gaussian isotropic'. **[refine]**

_refine.ls_goodness_of_fit_all (*float, su*)
_refine_ls_goodness_of_fit_all (*cif_core.dic 2.0.1*)
The least-squares goodness-of-fit parameter S for all data after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also the definition of **_refine.ls_restrained_S_all**.

$$S = \left(\frac{\sum \left| w | Y_{\rm obs} - Y_{\rm calc} |^2 \right|}{N_{\rm ref} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine.ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine.ls_structure_factor_coef**), w = the least-squares reflection weight [1/(e.s.d. squared)], $N_{\rm ref}$ = the number of reflections used in the refinement and $N_{\rm param}$ = the number of refined parameters; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_goodness_of_fit_all_esd** (associated esd).
[refine]

_refine.ls_goodness_of_fit_all_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.ls_goodness_of_fit_all**.
Related item: **_refine.ls_goodness_of_fit_all** (associated value). **[refine]**

_refine.ls_goodness_of_fit_gt (*float*)
_refine_ls_goodness_of_fit_gt (*cif_core.dic 2.3*)
The least-squares goodness-of-fit parameter S for significantly intense reflections (see **_reflns.threshold_expression**) after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also **_refine.ls_restrained_S_** definitions.

$$S = \left(\frac{\sum \left| w | Y_{\rm obs} - Y_{\rm calc} |^2 \right|}{N_{\rm ref} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine.ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine.ls_structure_factor_coef**), w = the least-squares reflection weight $[1/(u^2)]$, u = the standard uncertainty, $N_{\rm ref}$ = the number of reflections used in the refinement and $N_{\rm param}$ = the number of refined parameters; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_goodness_of_fit_obs** (alternate). **[refine]**

_refine.ls_goodness_of_fit_obs (*float, su*)
_refine_ls_goodness_of_fit_obs (*cif_core.dic 2.0.1*)
The least-squares goodness-of-fit parameter S for reflection data classified as 'observed' (see **_reflns.observed_criterion**) after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also the definition of **_refine.ls_restrained_S_obs**.

$$S = \left(\frac{\sum |w| Y_{\text{obs}} - Y_{\text{calc}}|^2|}{N_{\text{ref}} - N_{\text{param}}} \right)^{1/2},$$

where Y_{obs} = the observed coefficients (see **_refine.ls_structure_factor_coef**), Y_{calc} = the calculated coefficients (see **_refine.ls_structure_factor_coef**), w = the least-squares reflection weight [1/(e.s.d. squared)], N_{ref} = the number of reflections used in the refinement and N_{param} = the number of refined parameters; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_goodness_of_fit_obs_esd** (associated esd).
[refine]

_refine.ls_goodness_of_fit_obs_esd (*float*)
The standard uncertainty (estimated standard deviation) of **_refine.ls_goodness_of_fit_obs**.
Related item: **_refine.ls_goodness_of_fit_obs** (associated value). **[refine]**

_refine.ls_goodness_of_fit_ref (*float*)
_refine_ls_goodness_of_fit_ref (*cif_core.dic 2.3*)
The least-squares goodness-of-fit parameter S for all reflections included in the refinement after the final cycle of refinement. Ideally, account should be taken of parameters restrained in the least-squares refinement. See also **_refine.ls_restrained_S_** definitions.

$$S = \left(\frac{\sum |w| Y_{\text{obs}} - Y_{\text{calc}}|^2|}{N_{\text{ref}} - N_{\text{param}}} \right)^{1/2},$$

where Y_{obs} = the observed coefficients (see **_refine.ls_structure_factor_coef**), Y_{calc} = the calculated coefficients (see **_refine.ls_structure_factor_coef**), w = the least-squares reflection weight $[1/(u^2)]$, u = the standard uncertainty, N_{ref} = the number of reflections used in the refinement and N_{param} = the number of refined parameters; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_hydrogen_treatment (*ucode*)
_refine_ls_hydrogen_treatment (*cif_core.dic 2.0.1*)
Treatment of hydrogen atoms in the least-squares refinement.
The data value must be one of the following:

refall	refined all H-atom parameters
refxyz	refined H-atom coordinates only
refU	refined H-atom U's only
noref	no refinement of H-atom parameters
constr	H-atom parameters constrained
mixed	some constrained, some independent
undef	H-atom parameters not defined

[refine]

_refine.ls_matrix_type (*ucode*)
_refine_ls_matrix_type (*cif_core.dic 2.0.1*)
Type of matrix used to accumulate the least-squares derivatives.
The data value must be one of the following:

full	full
fullcycle	full with fixed elements per cycle
atomblock	block diagonal per atom
userblock	user-defined blocks
diagonal	diagonal elements only
sparse	selected elements only

[refine]

_refine.ls_number_constraints (*int*)
_refine_ls_number_constraints (*cif_core.dic 2.0.1*)
The number of constrained (non-refined or dependent) parameters in the least-squares process. These may be due to symmetry or any other constraint process (*e.g.* rigid-body refinement). See also **_atom_site.constraints** and **_atom_site.refinement_flags**. A general description of constraints may appear in **_refine.details**.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_number_parameters (*int*)
_refine_ls_number_parameters (*cif_core.dic 2.0.1*)
The number of parameters refined in the least-squares process. If possible, this number should include some contribution from the restrained parameters. The restrained parameters are distinct from the constrained parameters (where one or more parameters are linearly dependent on the refined value of another). Least-squares restraints often depend on geometry or energy considerations and this makes their direct contribution to this number, and to the goodness-of-fit calculation, difficult to assess.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_number_reflns_all (*int*)
The number of reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low**.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_number_reflns_obs (*int*)
_refine_ls_number_reflns (*cif_core.dic 2.0.1*)
The number of reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_number_reflns_R_free (*int*)
The number of reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_number_reflns_R_work (*int*)
The number of reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_number_restraints (*int*)
_refine_ls_number_restraints (*cif_core.dic 2.0.1*)
The number of restrained parameters. These are parameters which are not directly dependent on another refined parameter. Restrained parameters often involve geometry or energy dependencies. See also **_atom_site.constraints** and **_atom_site.refinement_flags**. A general description of refinement constraints may appear in **_refine.details**.
The permitted range is $[0, \infty)$. **[refine]**

_refine.ls_percent_reflns_obs (*float*)
The number of reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, expressed as a percentage of the number of geometrically observable reflections that satisfy the resolution limits.
[refine]

_refine.ls_percent_reflns_R_free (*float*)
The number of reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' *R* factor, expressed as a percentage of the number of geometrically observable reflections that satisfy the resolution limits.
[refine]

_refine.ls_R_factor_all (*float*)
_refine_ls_R_factor_all (*cif_core.dic 2.0.1*)
Residual factor *R* for all reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_wR_factor_all** (alternate). **[refine]**

_refine.ls_R_factor_gt (*float*)
_refine_ls_R_factor_gt (*cif_core.dic 2.3*)
Residual factor for the reflections (with number given by **_reflns.number_gt**) judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns.threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low**. This is the conventional *R* factor. See also **_refine.ls_wR_factor_** definitions.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_R_factor_obs** (alternate). **[refine]**

_refine.ls_R_factor_obs (*float*)
_refine_ls_R_factor_obs (*cif_core.dic 2.0.1*)
Residual factor *R* for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**. **_refine.ls_R_factor_obs** should not be confused with **_refine.ls_R_factor_R_work**; the former reports the results of a refinement in which all observed reflections were used, the latter a refinement in which a subset of the observed reflections were excluded from refinement for the calculation of a 'free' *R* factor. However, it would be meaningful to quote both values if a 'free' *R* factor were calculated for most of the refinement, but all of the observed reflections were used in the final rounds of refinement; such a protocol should be explained in **_refine.details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_wR_factor_obs** (alternate). **[refine]**

_refine.ls_R_factor_R_free (*float*)
Residual factor *R* for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' *R* factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related items: **_refine.ls_wR_factor_R_free** (alternate), **_refine.ls_R_factor_R_free_error** (associated error). **[refine]**

_refine.ls_R_factor_R_free_error (*float*)
The estimated error in **_refine.ls_R_factor_R_free**. The method used to estimate the error is described in the item **_refine.ls_R_factor_R_free_error_details**.
Related item: **_refine.ls_R_factor_R_free** (associated value). **[refine]**

_refine.ls_R_factor_R_free_error_details (*text*)
Special aspects of the method used to estimate the error in **_refine.ls_R_factor_R_free**.
[refine]

_refine.ls_R_factor_R_work (*float*)
Residual factor R for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**. **_refine.ls_R_factor_obs** should not be confused with **_refine.ls_R_factor_R_work**; the former reports the results of a refinement in which all observed reflections were used, the latter a refinement in which a subset of the observed reflections were excluded from refinement for the calculation of a 'free' R factor. However, it would be meaningful to quote both values if a 'free' R factor were calculated for most of the refinement, but all of the observed reflections were used in the final rounds of refinement; such a protocol should be explained in **_refine.details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_wR_factor_R_work** (alternate). **[refine]**

_refine.ls_R_Fsqd_factor_obs (*float*)
_refine_ls_R_Fsqd_factor (*cif_core.dic 2.0.1*)
Residual factor $R(F^2)$ for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, calculated on the squares of the observed and calculated structure-factor amplitudes.

$$R(F^2) = \frac{\sum |F^2_{\rm obs} - F^2_{\rm calc}|}{\sum |F^2_{\rm obs}|},$$

where $F^2_{\rm obs}$ = squares of the observed structure-factor amplitudes, $F^2_{\rm calc}$ = squares of the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_R_I_factor_obs (*float*)
_refine_ls_R_I_factor (*cif_core.dic 2.0.1*)
Residual factor $R(I)$ for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, calculated on the estimated reflection intensities. This is most often calculated in Rietveld refinements against powder data, where it is referred to as R_B or $R_{\rm Bragg}$.

$$R(I) = \frac{\sum |I_{\rm obs} - I_{\rm calc}|}{\sum |I_{\rm obs}|},$$

where $I_{\rm obs}$ = the net observed intensities, $I_{\rm calc}$ = the net calculated intensities and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_redundancy_reflns_all (*float*)
The ratio of the total number of observations of the reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** to the number of crystallographically unique reflections that satisfy the same limits.
[refine]

_refine.ls_redundancy_reflns_obs (*float*)
The ratio of the total number of observations of the reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion** to the number of crystallographically unique reflections that satisfy the same limits.
[refine]

_refine.ls_restrained_S_all (*float*)
_refine_ls_restrained_S_all (*cif_core.dic 2.0.1*)
The least-squares goodness-of-fit parameter S' for all reflections after the final cycle of least-squares refinement. This parameter explicitly includes the restraints applied in the least-squares process. See also the definition of **_refine.ls_goodness_of_fit_all**.

$$S' = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right| + \sum_r \left| w_r |P_{\rm calc} - P_{\rm targ}|^2 \right|}{N_{\rm ref} + N_{\rm restr} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine.ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine.ls_structure_factor_coef**), w = the least-squares reflection weight [1/(e.s.d. squared)], $P_{\rm calc}$ = the calculated restraint values, $P_{\rm targ}$ = the target restraint values, w_r = the restraint weight, $N_{\rm ref}$ = the number of reflections used in the refinement (see **_refine.ls_number_reflns_obs**), $N_{\rm restr}$ = the number of restraints (see **_refine.ls_number_restraints**) and $N_{\rm param}$ = the number of refined parameters (see **_refine.ls_number_parameters**); the sum $\sum$ is taken over the specified reflections and the sum $\sum_r$ is taken over the restraints.
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_restrained_S_obs (*float*)
_refine_ls_restrained_S_obs (*cif_core.dic 2.0.1*)
The least-squares goodness-of-fit parameter S' for reflection data classified as observed (see **_reflns.observed_criterion**) after the final cycle of least-squares refinement. This parameter explicitly includes the restraints applied in the least-squares process. See also the definition of **_refine.ls_goodness_of_fit_obs**.

$$S' = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right| + \sum_r \left| w_r |P_{\rm calc} - P_{\rm targ}|^2 \right|}{N_{\rm ref} + N_{\rm restr} - N_{\rm param}} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed coefficients (see **_refine.ls_structure_factor_coef**), $Y_{\rm calc}$ = the calculated coefficients (see **_refine.ls_structure_factor_coef**), w = the least-squares reflection weight [1/(e.s.d. squared)], $P_{\rm calc}$ = the calculated restraint values, $P_{\rm targ}$ = the target restraint values, w_r = the restraint weight, $N_{\rm ref}$ = the number of reflections used in the refinement (see **_refine.ls_number_reflns_obs**), $N_{\rm restr}$ = the number of restraints (see **_refine.ls_number_restraints**) and $N_{\rm param}$ = the number of refined parameters (see **_refine.ls_number_parameters**); the sum $\sum$ is taken over the specified reflections and the sum $\sum_r$ is taken over the restraints.
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_shift_over_esd_max (*float*)
_refine_ls_shift/esd_max (*cif_core.dic 2.0.1*)
The largest ratio of the final least-squares parameter shift to the final standard uncertainty (estimated standard deviation).
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_shift_over_esd_mean (*float*)
_refine_ls_shift/esd_mean (*cif_core.dic 2.0.1*)
The average ratio of the final least-squares parameter shift to the final standard uncertainty (estimated standard deviation).
The permitted range is $[0.0, \infty)$. **[refine]**

_refine.ls_shift_over_su_max (*float*)
_refine_ls_shift/su_max (*cif_core.dic 2.3*)
The largest ratio of the final least-squares parameter shift to the final standard uncertainty.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_shift_over_esd_max** (alternate). **[refine]**

_refine.ls_shift_over_su_max_lt (*float*)
_refine_ls_shift/su_max_lt (*cif_core.dic 2.3*)
An upper limit for the largest ratio of the final least-squares parameter shift to the final standard uncertainty. This item is used when the largest value of the shift divided by the final standard uncertainty is too small to measure.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_shift_over_su_max** (alternate). **[refine]**

_refine.ls_shift_over_su_mean (*float*)
_refine_ls_shift/su_mean (*cif_core.dic 2.3*)
The average ratio of the final least-squares parameter shift to the final standard uncertainty.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_shift_over_esd_mean** (alternate). **[refine]**

_refine.ls_shift_over_su_mean_lt (*float*)
_refine_ls_shift/su_mean_lt (*cif_core.dic 2.3*)
An upper limit for the average ratio of the final least-squares parameter shift to the final standard uncertainty. This item is used when the average value of the shift divided by the final standard uncertainty is too small to measure.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_shift_over_su_mean** (alternate). **[refine]**

_refine.ls_structure_factor_coef (*ucode*)
_refine_ls_structure_factor_coef (*cif_core.dic 2.0.1*)
Structure-factor coefficient $|F|$, F^2 or I used in the least-squares refinement process.
The data value must be one of the following:

F	structure-factor magnitude
Fsqd	structure factor squared
Inet	net intensity

[refine]

_refine.ls_weighting_details (*text*)
_refine_ls_weighting_details (*cif_core.dic 2.0.1*)
A description of special aspects of the weighting scheme used in least-squares refinement. Used to describe the weighting when the value of **_refine.ls_weighting_scheme** is specified as 'calc'.
Example:

```
; Sigdel model of Konnert-Hendrickson:
  Sigdel =
  Afsig +  Bfsig*(sin(theta)/lambda-1/6)
  Afsig = 22.0, Bfsig = 150.0
  at the beginning of refinement.
  Afsig = 16.0, Bfsig =  60.0
  at the end of refinement.
;
```

[refine]

_refine.ls_weighting_scheme (*ucode*)
_refine_ls_weighting_scheme (*cif_core.dic 2.0.1*)
The weighting scheme applied in the least-squares process. The standard code may be followed by a description of the weight (but see **_refine.ls_weighting_details** for a preferred approach).
The data value must be one of the following:

sigma	based on measured e.s.d.'s
unit	unit or no weights applied
calc	calculated weights applied

[refine]

_refine.ls_wR_factor_all (*float*)
_refine_ls_wR_factor_all (*cif_core.dic 2.0.1*)
Weighted residual factor *wR* for all reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_R_factor_all** (alternate). **[refine]**

_refine.ls_wR_factor_obs (*float*)
_refine_ls_wR_factor_obs (*cif_core.dic 2.0.1*)
Weighted residual factor *wR* for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_R_factor_obs** (alternate). **[refine]**

_refine.ls_wR_factor_R_free (*float*)
Weighted residual factor *wR* for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' *R* factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine.ls_R_factor_R_free** (alternate). **[refine]**

_refine.ls_wR_factor_R_work (*float*)

Weighted residual factor wR for reflections that satisfy the resolution limits established by **_refine.ls_d_res_high** and **_refine.ls_d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$wR = \left(\frac{\sum |w|Y_{\rm obs} - Y_{\rm calc}|^2|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$.

Related item: **_refine.ls_R_factor_R_work** (alternate). **[refine]**

_refine.occupancy_max (*float*)

The maximum value for occupancy found in the coordinate set.

The permitted range is $[0.0, \infty)$. **[refine]**

_refine.occupancy_min (*float*)

The minimum value for occupancy found in the coordinate set.

The permitted range is $[0.0, \infty)$. **[refine]**

_refine.overall_FOM_free_R_set (*float*)

_refine.ebi_overall_FOM_free_Rset (*ebi_extensions 1.0*)

Average figure of merit of phases of reflections not included in the refinement. This value is derived from the likelihood function

$$\mathrm{FOM} = I_1(X)/I_0(X),$$

where I_0, I_1 = zero- and first-order modified Bessel functions of the first kind, $X = \sigma_A|E_o||E_c|/\mathrm{SIGMA}$, E_o, E_c = normalized observed and calculated structure factors, $\sigma_A = \langle \cos 2\pi s\delta_x \rangle \sqrt{(\Sigma_P/\Sigma_N)}$ estimated using maximum likelihood, $\Sigma_P = \sum_{\text{atoms in model}} f^2$, $\Sigma_N = \sum_{\text{atoms in crystal}} f^2$, f = form factor of atoms, δ_x = expected error, $\mathrm{SIGMA} = (\sigma_{\{\mathrm{E;exp}\}})^2 + \varepsilon[1 - (\sigma_A)^2]$, $\sigma_{\{\mathrm{E;exp}\}}$ = uncertainties of normalized observed structure factors and ε = multiplicity of the diffracting plane.

Reference: Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst.* D**53**, 240–255.

[refine]

_refine.overall_FOM_work_R_set (*float*)

_refine.ebi_overall_FOM_work_Rset (*ebi_extensions 1.0*)

Average figure of merit of phases of reflections included in the refinement. This value is derived from the likelihood function

$$\mathrm{FOM} = I_1(X)/I_0(X),$$

where I_0, I_1 = zero- and first-order modified Bessel functions of the first kind, $X = \sigma_A|E_o||E_c|/\mathrm{SIGMA}$, E_o, E_c = normalized observed and calculated structure factors, $\sigma_A = \langle \cos 2\pi s\delta_x \rangle \sqrt{(\Sigma_P/\Sigma_N)}$ estimated using maximum likelihood, $\Sigma_P = \sum_{\text{atoms in model}} f^2$, $\Sigma_N = \sum_{\text{atoms in crystal}} f^2$, f = form factor of atoms, δ_x = expected error, $\mathrm{SIGMA} = (\sigma_{\{\mathrm{E;exp}\}})^2 + \varepsilon[1 - (\sigma_A)^2]$, $\sigma_{\{\mathrm{E;exp}\}}$ = uncertainties of normalized observed structure factors and ε = multiplicity of the diffracting plane.

Reference: Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst.* D**53**, 240–255.

[refine]

_refine.overall_SU_B (*float*)

_refine.ebi_Overall_ESU_B (*ebi_extensions 1.0*)

The overall standard uncertainty (estimated standard deviation) of the displacement parameters based on a maximum-likelihood residual. The overall standard uncertainty $(\sigma_B)^2$ gives an idea of the uncertainty in the B values of averagely defined atoms (atoms with B values equal to the average B value).

$$(\sigma_B)^2 = \frac{8N_a}{\sum_i [1/\Sigma - (E_o)^2(1-m^2)](\mathrm{SUM\_AS})s^4},$$

where $\mathrm{SUM\_AS} = (\sigma_A)^2/\Sigma^2$, N_a = number of atoms, $\Sigma = (\sigma_{\{E;\mathrm{exp}\}})^2 + \varepsilon[1 - (\sigma_A)^2]$, E_o = normalized structure factors, $\sigma_{\{E;\mathrm{exp}\}}$ = experimental uncertainties of normalized structure factors, $\sigma_A = \langle \cos 2\pi s\delta_x \rangle \sqrt{(\Sigma_P/\Sigma_N)}$ estimated using maximum likelihood, $\Sigma_P = \sum_{\text{atoms in model}} f^2$, $\Sigma_N = \sum_{\text{atoms in crystal}} f^2$, f = form factor of atoms, δ_x = expected error, m = figure of merit of phases of reflections included in the summation, s = reciprocal-space vector and ε = multiplicity of the diffracting plane; the summation is over all reflections included in refinement.

References: σ_A estimation: Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst.* D**53**, 240–255. SU ML estimation: Murshudov, G. N. & Dodson, E. J. (1997). *CCP4 Newsletter on Protein Crystallography*, No. 33, January 1997, pp. 31–39. (http://www.ccp4.ac.uk/newsletters/newsletter33/murshudov.html)

[refine]

_refine.overall_SU_ML (*float*)

_refine.ebi_Overall_ESU_ML (*ebi_extensions 1.0*)

The overall standard uncertainty (estimated standard deviation) of the positional parameters based on a maximum-likelihood residual. The overall standard uncertainty $(\sigma_X)^2$ gives an idea of the uncertainty in the position of averagely defined atoms (atoms with B values equal to the average B value).

$$(\sigma_X)^2 = \frac{3N_a}{8\pi^2 \sum_i [1/\Sigma - (E_o)^2(1-m^2)](\mathrm{SUM\_AS})s^2},$$

where $\mathrm{SUM\_AS} = (\sigma_A)^2/\Sigma^2$, N_a = number of atoms, $\Sigma = (\sigma_{\{E;\mathrm{exp}\}})^2 + \varepsilon[1 - (\sigma_A)^2]$, E_o = normalized structure factors, $\sigma_{\{E;\mathrm{exp}\}}$ = experimental uncertainties of normalized structure factors, $\sigma_A = \langle \cos 2\pi s\delta_x \rangle \sqrt{(\Sigma_P/\Sigma_N)}$ estimated using maximum likelihood, $\Sigma_P = \sum_{\text{atoms in model}} f^2$, $\Sigma_N = \sum_{\text{atoms in crystal}} f^2$, f = form factor of atoms, δ_x = expected error, m = figure of merit of phases of reflections included in the summation, s = reciprocal-space vector and ε = multiplicity of the diffracting plane; the summation is over all reflections included in refinement.

References: σ_A estimation: Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst.* D**53**, 240–255. SU ML estimation: Murshudov, G. N. & Dodson, E. J. (1997). *CCP4 Newsletter on Protein Crystallography*, No. 33, January 1997, pp. 31–39. (http://www.ccp4.ac.uk/newsletters/newsletter33/murshudov.html)

[refine]

_refine.overall_SU_R_Cruickshank_DPI (*float*)
_refine.ebi_Overall_ESU_R_Cruickshanks_DPI (*ebi_extensions 1.0*)
The overall standard uncertainty (estimated standard deviation) of the displacement parameters based on the crystallographic *R* value, expressed in a formalism known as the dispersion precision indicator (DPI). The overall standard uncertainty (σ_B) gives an idea of the uncertainty in the *B* values of averagely defined atoms (atoms with *B* values equal to the average *B* value).

$$(\sigma_B)^2 = 0.65\frac{N_a}{(N_o - N_p)}(R_{\text{value}})^2(D_{\text{min}})^2C^{(-2/3)},$$

where N_a = number of atoms, N_o = number of reflections included in refinement, N_p = number of refined parameters, R_{value} = conventional crystallographic *R* value, D_{min} = maximum resolution and *C* = completeness of data.

References: Cruickshank, D. W. J. (1999). *Acta Cryst.* D**55**, 583–601; Murshudov, G. N. & Dodson, E. J. (1997). *CCP4 Newsletter on Protein Crystallography*, No. 33, January 1997, pp. 31–39. (http://www.ccp4.ac.uk/newsletters/newsletter33/murshudov.html)

[refine]

_refine.overall_SU_R_free (*float*)
_refine.ebi_Overall_ESU_Rfree (*ebi_extensions 1.0*)
The overall standard uncertainty (estimated standard deviation) of the displacement parameters based on the free *R* value. The overall standard uncertainty gives an idea of the uncertainty in the *B* values of averagely defined atoms (atoms with *B* values equal to the average *B* value).

$$(\sigma_B)^2 = 0.65\frac{N_a}{N_o}(R_{\text{free}})^2(D_{\text{min}})^2C^{(-2/3)},$$

where N_a = number of atoms, N_o = number of reflections included in the refinement, R_{free} = conventional free crystallographic *R* value calculated using the reflections not included in the refinement, D_{min} = maximum resolution and *C* = completeness of data.

References: Cruickshank, D. W. J. (1999). *Acta Cryst.* D**55**, 583–601; Murshudov, G. N. & Dodson, E. J. (1997). *CCP4 Newsletter on Protein Crystallography*, No. 33, January 1997, pp. 31–39. (http://www.ccp4.ac.uk/newsletters/newsletter33/murshudov.html)

[refine]

_refine.solvent_model_details (*text*)
Special aspects of the solvent model used during refinement.

[refine]

_refine.solvent_model_param_bsol (*float*)
The value of the B_{SOL} solvent-model parameter describing the average isotropic displacement parameter of disordered solvent atoms. This is one of the two parameters (the other is **_refine.solvent_model_param_ksol**) in Tronrud's method of modelling the contribution of bulk solvent to the scattering. The standard scale factor is modified according to the expression

$$k_0 \exp(-B_0 s^2)[1 - K_{\text{SOL}} \exp(-B_{\text{SOL}} s^2)],$$

where k_0 and B_0 are the scale factors for the protein.

Reference: Tronrud, D. E. (1997). *Methods Enzymol.* **277**, 243–268.

[refine]

_refine.solvent_model_param_ksol (*float*)
The value of the K_{SOL} solvent-model parameter describing the ratio of the electron density in the bulk solvent to the electron density in the molecular solute. This is one of the two parameters (the other is **_refine.solvent_model_param_bsol**) in Tronrud's method of modelling the contribution of bulk solvent to the scattering. The standard scale factor is modified according to the expression

$$k_0 \exp(-B_0 s^2)[1 - K_{\text{SOL}} \exp(-B_{\text{SOL}} s^2)],$$

where k_0 and B_0 are the scale factors for the protein.

Reference: Tronrud, D. E. (1997). *Methods Enzymol.* **277**, 243–268.

[refine]

REFINE_ANALYZE

Data items in the REFINE_ANALYZE category record details about the refined structure that are often used to analyze the refinement and assess its quality. A given computer program may or may not produce values corresponding to these data names.
Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine_analyze.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_refine_analyze.entry_id
_refine_analyze.Luzzati_coordinate_error_obs
_refine_analyze.Luzzati_d_res_low_obs
        5HVP 0.056  2.51
```

* **_refine_analyze.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

[refine_analyze]

_refine_analyze.Luzzati_coordinate_error_free (*float*)
The estimated coordinate error obtained from the plot of the *R* value *versus* $\sin(\theta)/\lambda$ for the reflections treated as a test set during refinement.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_coordinate_error_obs (*float*)
The estimated coordinate error obtained from the plot of the *R* value *versus* $\sin(\theta)/\lambda$ for reflections classified as observed.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_d_res_low_free (*float*)
The value of the low-resolution cutoff used in constructing the Luzzati plot for reflections treated as a test set during refinement.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_d_res_low_obs (*float*)
The value of the low-resolution cutoff used in constructing the Luzzati plot for reflections classified as observed.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_sigma_a_free (*float*)
The value of σ_a used in constructing the Luzzati plot for the reflections treated as a test set during refinement. Details of the estimation of σ_a can be specified in **_refine_analyze.Luzzati_sigma_a_free_details**.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_sigma_a_free_details (*text*)
Details of the estimation of σ_a for the reflections treated as a test set during refinement.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_sigma_a_obs (*float*)
The value of σ_a used in constructing the Luzzati plot for reflections classified as observed. Details of the estimation of σ_a can be specified in **_refine_analyze.Luzzati_sigma_a_obs_details**.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.Luzzati_sigma_a_obs_details (*text*)
Special aspects of the estimation of σ_a for the reflections classified as observed.

Reference: Luzzati, V. (1952). *Acta Cryst.* **5**, 802–810.

[refine_analyze]

_refine_analyze.number_disordered_residues (*float*)
The number of discretely disordered residues in the refined model.

[refine_analyze]

_refine_analyze.occupancy_sum_hydrogen (*float*)
The sum of the occupancies of the hydrogen atoms in the refined model.

[refine_analyze]

_refine_analyze.occupancy_sum_non_hydrogen (*float*)
The sum of the occupancies of the non-hydrogen atoms in the refined model.

[refine_analyze]

_refine_analyze.RG_d_res_high (*float*)
_refine_analyze.ebi_RG_d_res_high (*ebi_extensions 1.0*)
The value of the high-resolution cutoff in ångströms used in the calculation of the Hamilton generalized R factor (R_G) stored in **_refine_analyze.RG_work** and **_refine_analyze.RG_free**.

Reference: Hamilton, W. C. (1965). *Acta Cryst.* **18**, 502–510.

The permitted range is $[0.0, \infty)$. **[refine_analyze]**

_refine_analyze.RG_d_res_low (*float*)
_refine_analyze.ebi_RG_d_res_low (*ebi_extensions 1.0*)
The value of the low-resolution cutoff in ångströms used in the calculation of the Hamilton generalized R factor (R_G) stored in **_refine_analyze.RG_work** and **_refine_analyze.RG_free**.

Reference: Hamilton, W. C. (1965). *Acta Cryst.* **18**, 502–510.

The permitted range is $[0.0, \infty)$. **[refine_analyze]**

_refine_analyze.RG_free (*float*)
_refine_analyze.ebi_RG_free (*ebi_extensions 1.0*)
The Hamilton generalized R factor for all reflections that satisfy the resolution limits established by **_refine_analyze.RG_d_res_high** and **_refine_analyze.RG_d_res_low** for the free R set of reflections that were excluded from the refinement.

$$R_G = \sqrt{\frac{\sum_i \sum_j w_{i,j}(|F_{\rm obs}|_i - G|F_{\rm calc}|_i)(|F_{\rm obs}|_j - G|F_{\rm calc}|_j)}{\sum_i \sum_j w_{i,j}|F_{\rm obs}|_i|F_{\rm obs}|_j}},$$

where $|F_{\rm obs}|$ = the observed structure-factor amplitudes, $|F_{\rm calc}|$ = the calculated structure-factor amplitudes, G = the scale factor which puts $|F_{\rm calc}|$ on the same scale as $|F_{\rm obs}|$ and $w_{i,j}$ = the weight for the combination of the reflections i and j; $\sum_i$ and $\sum_j$ are taken over the specified reflections.

When the covariance of the amplitudes of reflection i and reflection j is zero (*i.e.* the reflections are independent), $w_{i,i}$ can be redefined as w_i and the nested sums collapsed into one sum:

$$R_G = \sqrt{\frac{\sum_i w_i(|F_{\rm obs}|_i - G|F_{\rm calc}|_i)^2}{\sum_i w_i|F_{\rm obs}|_i^2}}.$$

Reference: Hamilton, W. C. (1965). *Acta Cryst.* **18**, 502–510.

The permitted range is $[0.0, \infty)$. **[refine_analyze]**

_refine_analyze.RG_free_work_ratio (*float*)
_refine_analyze.ebi_RG_work_free_ratio (*ebi_extensions 1.0*)
The observed ratio of $R_{G\rm free}$ to $R_{G\rm work}$. The expected R_G ratio is the value that should be achievable at the end of a structure refinement when only random uncorrelated errors exist in the data and the model provided that the observations are properly weighted. When compared with the observed R_G ratio, it may indicate that a structure has not reached convergence or a model has been overrefined with no corresponding improvement in the model.

In an unrestrained refinement, the ratio of $R_{G\rm free}$ to $R_{G\rm work}$ with only random uncorrelated errors at convergence depends only on the number of reflections and the number of parameters according to

$$\sqrt{(f+m)/(f-m)},$$

where f = the number of included structure amplitudes and target distances and m = the number of parameters being refined.

In the restrained case, $R_{G\rm free}$ is calculated from a random selection of residuals including both the structure amplitudes and the restraints. When restraints are included in the refinement, the R_G ratio requires a term for the contribution to the minimized residual at convergence, $D_{\rm restr}$, due to those restraints:

$$D_{\rm restr} = r - \sum[w_i \cdot (a_i)^t \cdot (H)^{-1} a_i],$$

where r is the number of geometrical, displacement-parameter and other restraints, H is the (m,m) normal matrix given by $A^t \cdot W \cdot A$, W is the (n,n) symmetric weight matrix of the included observations, A is the least-squares design matrix of derivatives of order (n,m) and a_i is the ith row of A. Then the expected R_G ratio becomes

$$\sqrt{[f + (m - r + D_{\rm restr})]/[f - (m - r + D_{\rm restr})]}.$$

There is no data name for the expected value of $R_{G\rm free}/R_{G\rm work}$ yet.

Reference: Tickle, I. J., Laskowski, R. A. & Moss, D. S. (1998). *Acta Cryst.* D**54**, 547–557.

The permitted range is $[0.0, \infty)$. **[refine_analyze]**

_refine_analyze.RG_work (*float*)
_refine_analyze.ebi_RG_work (*ebi_extensions 1.0*)
The Hamilton generalized R factor for all reflections that satisfy the resolution limits established by **_refine_analyze.RG_d_res_high** and **_refine_analyze.RG_d_res_low** and for those reflections included in the working set when a free R set of reflections is omitted from the refinement.

$$R_G = \sqrt{\frac{\sum_i \sum_j w_{i,j}(|F_{\rm obs}|_i - G|F_{\rm calc}|_i)(|F_{\rm obs}|_j - G|F_{\rm calc}|_j)}{\sum_i \sum_j w_{i,j}|F_{\rm obs}|_i|F_{\rm obs}|_j}},$$

where $|F_{\rm obs}|$ = the observed structure-factor amplitudes, $|F_{\rm calc}|$ = the calculated structure-factor amplitudes, G = the scale factor which puts $|F_{\rm calc}|$ on the same scale as $|F_{\rm obs}|$ and $w_{i,j}$ = the weight for the combination of the reflections i and j; $\sum_i$ and $\sum_j$ are taken over the specified reflections.

When the covariance of the amplitudes of reflection i and reflection j is zero (*i.e.* the reflections are independent), $w_{i,i}$ can be redefined as w_i and the nested sums collapsed into one sum:

$$R_G = \sqrt{\frac{\sum_i w_i(|F_{\rm obs}|_i - G|F_{\rm calc}|_i)^2}{\sum_i w_i|F_{\rm obs}|_i^2}}.$$

Reference: Hamilton, W. C. (1965). *Acta Cryst.* **18**, 502–510.
The permitted range is $[0.0, \infty)$. **[refine_analyze]**

REFINE_B_ISO

Data items in the REFINE_B_ISO category record details about the treatment of isotropic B factors (displacement parameters) during refinement.
Category group(s): **inclusive_group**, **refine_group**
Category key(s): **_refine_B_iso.class**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_refine_B_iso.class
_refine_B_iso.treatment
 'protein'    isotropic
 'solvent'    isotropic
 'inhibitor'  isotropic
```

*_refine_B_iso.class (*text*)
A class of atoms treated similarly for isotropic B-factor (displacement-parameter) refinement.
Examples: 'all', 'protein', 'solvent', 'sugar-phosphate backbone'.
[refine_B_iso]

_refine_B_iso.details (*text*)
A description of special aspects of the isotropic B-factor (displacement-parameter) refinement for the class of atoms described in **_refine_B_iso.class**.
Example:
```
; The temperature factors of atoms in the side chain of Arg 92
were held fixed due to unstable behavior in refinement.
;
```
[refine_B_iso]

_refine_B_iso.treatment (*ucode*)
The treatment of isotropic B-factor (displacement-parameter) refinement for a class of atoms defined in **_refine_B_iso.class**.
The data value must be one of the following:
fixed
isotropic
anisotropic
[refine_B_iso]

_refine_B_iso.value (*float*)
The value of the isotropic B factor (displacement parameter) assigned to a class of atoms defined in **_refine_B_iso.class**. Meaningful only for atoms with fixed isotropic B factors.
[refine_B_iso]

REFINE_FUNCT_MINIMIZED

Data items in the REFINE_FUNCT_MINIMIZED category record details about the individual terms of the function minimized during refinement.
Category group(s): **inclusive_group**, **refine_group**
Category key(s): **_refine_funct_minimized.type**

Example 1 – based on RESTRAIN refinement for the CCP4 test data set toxd.

```
loop_
_refine_funct_minimized.type
_refine_funct_minimized.number_terms
_refine_funct_minimized.residual
'sum(W*Delta(Amplitude)^2'                3009      1621.3
'sum(W*Delta(Plane+Rigid)^2'                85       56.68
'sum(W*Delta(Distance)^2'                 1219      163.59
'sum(W*Delta(U-tempfactors)^2'            1192      69.338
```

_refine_funct_minimized.number_terms (*int*)
_ebi_refine_funct_minimized.NumTerms (*ebi_extensions 1.0*)
The number of observations in this term. For example, if the term is a residual of the X-ray intensities, this item would contain the number of reflections used in the refinement.
The permitted range is $[0, \infty)$. **[refine_funct_minimized]**

_refine_funct_minimized.residual (*float*)
_ebi_refine_funct_minimized.Residual (*ebi_extensions 1.0*)
The residual for this term of the function that was minimized during the refinement.
The permitted range is $[0.0, \infty)$. **[refine_funct_minimized]**

*_refine_funct_minimized.type (*line*)
_ebi_refine_funct_minimized.type (*ebi_extensions 1.0*)
The type of the function being minimized.
[refine_funct_minimized]

_refine_funct_minimized.weight (*float*)
_ebi_refine_funct_minimized.weight (*ebi_extensions 1.0*)
The weight applied to this term of the function that was minimized during the refinement.
[refine_funct_minimized]

REFINE_HIST

Data items in the REFINE_HIST category record details about the steps during the refinement of the structure. These data items are not meant to be as thorough a description of the refinement as is provided for the final model in other categories; rather, these data items provide a mechanism for sketching out the progress of the refinement, supported by a small set of representative statistics.
Category group(s): **inclusive_group**, **refine_group**
Category key(s): **_refine_hist.cycle_id**

Example 1 – based on laboratory records for the collagen-like peptide [(POG)$_4$ EKG (POG)$_5$]$_3$.

```
_refine_hist.cycle_id                C134
_refine_hist.d_res_high              1.85
_refine_hist.d_res_low               20.0
_refine_hist.number_atoms_solvent     217
_refine_hist.number_atoms_total       808
_refine_hist.number_reflns_all       6174
_refine_hist.number_reflns_obs       4886
_refine_hist.number_reflns_R_free     476
_refine_hist.number_reflns_R_work    4410
_refine_hist.R_factor_all            .265
_refine_hist.R_factor_obs            .195
_refine_hist.R_factor_R_free         .274
_refine_hist.R_factor_R_work         .160
_refine_hist.details
; Add majority of solvent molecules. B factors refined by
  group. Continued to remove misplaced water molecules.
;
```

* **_refine_hist.cycle_id** (*code*)
The value of **_refine_hist.cycle_id** must uniquely identify a record in the REFINE_HIST list. Note that this item need not be a number; it can be any unique identifier.

[refine_hist]

* **_refine_hist.d_res_high** (*float*)
The lowest value for the interplanar spacings for the reflection data for this cycle of refinement. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[refine_hist]**

* **_refine_hist.d_res_low** (*float*)
The highest value for the interplanar spacings for the reflection data for this cycle of refinement. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[refine_hist]**

_refine_hist.details (*text*)
A description of special aspects of this cycle of the refinement process.
Example:
```
; Residues 13-17 fit and added to model; substantial
rebuilding of loop containing residues 43-48; addition of
first atoms to solvent model; ten cycles of Prolsq refinement.
;
```
[refine_hist]

_refine_hist.number_atoms_solvent (*int*)
The number of solvent atoms that were included in the model at this cycle of the refinement.
The permitted range is $[0, \infty)$. **[refine_hist]**

_refine_hist.number_atoms_total (*int*)
The total number of atoms that were included in the model at this cycle of the refinement.
The permitted range is $[0, \infty)$. **[refine_hist]**

_refine_hist.number_reflns_all (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low**.
The permitted range is $[0, \infty)$. **[refine_hist]**

_refine_hist.number_reflns_obs (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low** and the observation criterion established by **_reflns.observed_criterion**.
The permitted range is $[0, \infty)$. **[refine_hist]**

_refine_hist.number_reflns_R_free (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.
The permitted range is $[0, \infty)$. **[refine_hist]**

_refine_hist.number_reflns_R_work (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.
The permitted range is $[0, \infty)$. **[refine_hist]**

_refine_hist.R_factor_all (*float*)
Residual factor R for reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes and $F_{\rm calc}$ = the calculated structure-factor amplitudes; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[refine_hist]**

_refine_hist.R_factor_obs (*float*)
Residual factor R for reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low** and the observation criterion established by **_reflns.observed_criterion**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes and $F_{\rm calc}$ = the calculated structure-factor amplitudes; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[refine_hist]**

_refine_hist.R_factor_R_free (*float*)
Residual factor R for reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes and $F_{\rm calc}$ = the calculated structure-factor amplitudes; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$. **[refine_hist]**

_refine_hist.R_factor_R_work (*float*)

Residual factor R for reflections that satisfy the resolution limits established by **_refine_hist.d_res_high** and **_refine_hist.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes and $F_{\rm calc}$ = the calculated structure-factor amplitudes; the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$. **[refine_hist]**

REFINE_LS_CLASS

Data items in the REFINE_LS_CLASS category record details about the reflections used for the structure refinement for each reflection class separately.

Category key(s): **_refine_ls_class.code**

*Example 1 – data for a modulated structure from van Smaalen [J. Phys. Condens. Matter (1991), **3**, 1247–1263].*

```
loop_
  _refine_ls_class.R_factor_gt
  _refine_ls_class.code
      0.057    'Main'
      0.074    'Com'
      0.064    'NbRefls'
      0.046    'LaRefls'
      0.112    'Sat1'
      0.177    'Sat2'
```

* **_refine_ls_class.code** (*code*)

_refine_ls_class_code (*cif_core.dic 2.3*)

The code identifying a certain reflection class. This code must match a **_reflns_class.code**.

Examples: '1', 'm1', 's2'. **[refine_ls_class]**

_refine_ls_class.d_res_high (*float*)

_refine_ls_class_d_res_high (*cif_core.dic 2.3*)

For each reflection class, the lowest value in ångströms for the interplanar spacings for the reflections used in the refinement. This is called the highest resolution.

The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

_refine_ls_class.d_res_low (*float*)

_refine_ls_class_d_res_low (*cif_core.dic 2.3*)

For each reflection class, the highest value in ångströms for the interplanar spacings for the reflections used in the refinement. This is called the lowest resolution.

The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

_refine_ls_class.R_factor_all (*float*)

_refine_ls_class_R_factor_all (*cif_core.dic 2.3*)

For each reflection class, the residual factor for all reflections satisfying the resolution limits established by **_refine_ls_class.d_res_high** and **_refine_ls_class.d_res_low**. This is the conventional R factor. See also the definition of **_refine_ls_class.wR_factor_all**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.

The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

_refine_ls_class.R_factor_gt (*float*)

_refine_ls_class_R_factor_gt (*cif_core.dic 2.3*)

For each reflection class, the residual factor for significantly intense reflections (see **_reflns.threshold_expression**) included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_class.d_res_high** and **_refine_ls_class.d_res_low**. This is the conventional R factor. See also the definition of **_refine_ls_class.wR_factor_all**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.

The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

_refine_ls_class.R_Fsqd_factor (*float*)

_refine_ls_class_R_Fsqd_factor (*cif_core.dic 2.3*)

For each reflection class, the residual factor $R(F^2)$ calculated on the squared amplitudes of the observed and calculated structure factors for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns.threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_class.d_res_high** and **_refine_ls_class.d_res_low**.

$$R(F^2) = \frac{\sum |F^2_{\rm obs} - F^2_{\rm calc}|}{\sum |F^2_{\rm obs}|},$$

where $F^2_{\rm obs}$ = squares of the observed structure-factor amplitudes, $F^2_{\rm calc}$ = squares of the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.

The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

_refine_ls_class.R_I_factor (*float*)

_refine_ls_class_R_I_factor (*cif_core.dic 2.3*)

For each reflection class, the residual factor $R(I)$ for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns.threshold_expression**) and included in the refinement. This is most often calculated in Rietveld refinements against powder data, where it is referred to as R_B or $R_{\rm Bragg}$.

$$R(I) = \frac{\sum |I_{\rm obs} - I_{\rm calc}|}{\sum |I_{\rm obs}|},$$

where $I_{\rm obs}$ = the net observed intensities, $I_{\rm calc}$ = the net calculated intensities and the sum is taken over the reflections of this class.

The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

_refine_ls_class.wR_factor_all (*float*)
_refine_ls_class_wR_factor_all (*cif_core.dic 2.3*)
For each reflection class, the weighted residual factor for all reflections included in the refinement. The reflections also satisfy the resolution limits established by **_refine_ls_class.d_res_high** and **_refine_ls_class.d_res_low**. See also the **_refine_ls_class.R_factor_** definitions.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitudes specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitudes specified by **_refine.ls_structure_factor_coef**, w = the least-squares weight and the sum is taken over the reflections of this class.
The permitted range is $[0.0, \infty)$. **[refine_ls_class]**

REFINE_LS_RESTR

Data items in the REFINE_LS_RESTR category record details about the restraints applied to various classes of parameters during the least-squares refinement.
Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine_ls_restr.type**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_refine_ls_restr.type
_refine_ls_restr.dev_ideal_target
_refine_ls_restr.dev_ideal
_refine_ls_restr.number
_refine_ls_restr.criterion
_refine_ls_restr.rejects
 'p_bond_d'           0.020  0.018  1654  '> 2\s'  22
 'p_angle_d'          0.030  0.038  2246  '> 2\s'  139
 'p_planar_d'         0.040  0.043  498   '> 2\s'  21
 'p_planar'           0.020  0.015  270   '> 2\s'  1
 'p_chiral'           0.150  0.177  278   '> 2\s'  2
 'p_singtor_nbd'      0.500  0.216  582   '> 2\s'  0
 'p_multtor_nbd'      0.500  0.207  419   '> 2\s'  0
 'p_xyhbond_nbd'      0.500  0.245  149   '> 2\s'  0
 'p_planar_tor'       3.0    2.6    203   '> 2\s'  9
 'p_staggered_tor'    15.0   17.4   298   '> 2\s'  31
 'p_orthonormal_tor'  20.0   18.1   12    '> 2\s'  1
```

_refine_ls_restr.criterion (*text*)
A criterion used to define a parameter value that deviates significantly from its ideal value in the model obtained by restrained least-squares refinement.
Example: '> 3\s'. **[refine_ls_restr]**

_refine_ls_restr.dev_ideal (*float*)
For the given parameter type, the root-mean-square deviation between the ideal values used as restraints in the least-squares refinement and the values obtained by refinement. For instance, bond distances may deviate by 0.018 Å (r.m.s.) from ideal values in the current model.
The permitted range is $[0.0, \infty)$. **[refine_ls_restr]**

_refine_ls_restr.dev_ideal_target (*float*)
For the given parameter type, the target root-mean-square deviation between the ideal values used as restraints in the least-squares refinement and the values obtained by refinement.
The permitted range is $[0.0, \infty)$. **[refine_ls_restr]**

_refine_ls_restr.number (*int*)
The number of parameters of this type subjected to restraint in least-squares refinement.
The permitted range is $[0, \infty)$. **[refine_ls_restr]**

_refine_ls_restr.rejects (*int*)
The number of parameters of this type that deviate from ideal values by more than the amount defined in **_refine_ls_restr.criterion** in the model obtained by restrained least-squares refinement.
The permitted range is $[0, \infty)$. **[refine_ls_restr]**

***_refine_ls_restr.type** (*line*)
The type of the parameter being restrained. Explicit sets of data values are provided for the programs *PROTIN/PROLSQ* (beginning with *p_*) and *RESTRAIN* (beginning with *RESTRAIN_*). As computer programs change, these data values are given as examples, not as an enumeration list. Computer programs that convert a data block to a refinement table will expect the exact form of the data values given here to be used.
The following item(s) have an equivalent role in their respective categories:
_refine_ls_restr_type.type.
Examples: 'p_bond_d' (bond distance), 'p_angle_d' (bond angle expressed as a distance), 'p_planar_d' (planar 1,4 distance), 'p_xhbond_d' (*X*—H bond distance), 'p_xhangle_d' (*X*—H bond angle expressed as a distance), 'p_hydrog_d' (hydrogen distance), 'p_special_d' (special distance), 'p_planar' (planes), 'p_chiral' (chiral centres), 'p_singtor_nbd' (single-torsion non-bonded contact), 'p_multtor_nbd' (multiple-torsion non-bonded contact), 'p_xyhbond_nbd' (possible $X\cdots Y$ hydrogen bond), 'p_xhyhbond_nbd' (possible *X*—H$\cdots Y$ hydrogen bond), 'p_special_tor' (special torsion angle), 'p_planar_tor' (planar torsion angle), 'p_staggered_tor' (staggered torsion angle), 'p_orthonormal_tor' (orthonormal torsion angle), 'p_mcbond_it' (main-chain bond isotropic displacement parameter), 'p_mcangle_it' (main-chain angle isotropic displacement parameter), 'p_scbond_it' (side-chain bond isotropic displacement parameter), 'p_scangle_it' (side-chain angle isotropic displacement parameter), 'p_xhbond_it' (*X*—H bond isotropic displacement parameter), 'p_xhangle_it' (*X*—H angle isotropic displacement parameter), 'p_special_it' (special isotropic displacement parameter), 'RESTRAIN_Distances < 2.12' (the root-mean-square deviation of the difference between the values calculated from the structures used to compile the restraints dictionary parameters and the dictionary values themselves in the distance range less than 2.12 Å), 'RESTRAIN_Distances 2.12 < D < 2.625' (the root-mean-square deviation of the difference between the values calculated from the structures used to compile the restraints dictionary parameters and the dictionary values themselves in the distance range 2.12–2.625 Å), 'RESTRAIN_Distances > 2.625' (the root-mean-square deviation of the difference between the values calculated from the structures used to compile the restraints dictionary parameters and the dictionary values themselves in the distance range greater than 2.625 Å), 'RESTRAIN_Peptide Planes' (the root-mean-square deviation of the difference between the values calculated from the structures used to compile the restraints dictionary parameters and the dictionary values themselves for peptide planes), 'RESTRAIN_Ring and other planes' (the root-mean-square deviation of the difference between the values calculated from the structures used to compile the restraints dictionary parameters and the dictionary values themselves for rings and planes other than peptide planes), 'RESTRAIN_rms diffs for Uiso atoms at dist 1.2-1.4', 'RESTRAIN_rms diffs for Uiso atoms at dist 1.4-1.6', 'RESTRAIN_rms diffs for Uiso atoms at dist 1.8-2.0', 'RESTRAIN_rms diffs for Uiso atoms at dist 2.0-2.2', 'RESTRAIN_rms diffs for Uiso atoms at dist 2.2-2.4', 'RESTRAIN_rms diffs for Uiso atoms at dist >2.4'.
[refine_ls_restr]

_refine_ls_restr.weight (*float*)
The weighting value applied to this type of restraint in the least-squares refinement.
[refine_ls_restr]

REFINE_LS_RESTR_NCS

Data items in the REFINE_LS_RESTR_NCS category record details about the restraints applied to atom positions in domains related by noncrystallographic symmetry during least-squares refinement, and also about the deviation of the restrained atomic parameters at the end of the refinement. It is expected that these values will only be reported once for each set of restrained domains.

Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine_ls_restr_ncs.dom_id**

Example 1 – based on laboratory records for the collagen-like peptide, HYP-.

```
_refine_ls_restr_ncs.dom_id             d2
_refine_ls_restr_ncs.weight_position    300.0
_refine_ls_restr_ncs.weight_B_iso       2.0
_refine_ls_restr_ncs.rms_dev_position   0.09
_refine_ls_restr_ncs.rms_dev_B_iso      0.16
_refine_ls_restr_ncs.ncs_model_details
;
 NCS restraint for pseudo-twofold symmetry between domains
 d1 and d2.  Position weight coefficient given in
 Kcal/(mol \%A^2^) and isotropic B weight coefficient given
 in \%A^2^.
;
```

* **_refine_ls_restr_ncs.dom_id** (*code*)
This data item is a pointer to **_struct_ncs_dom.id** in the STRUCT_NCS_DOM category.

[refine_ls_restr_ncs]

_refine_ls_restr_ncs.ncs_model_details (*text*)
Special aspects of the manner in which noncrystallographic restraints were applied to atomic parameters in the domain specified by **_refine_ls_restr_ncs.dom_id** and equivalent atomic parameters in the domains against which it was restrained.

[refine_ls_restr_ncs]

_refine_ls_restr_ncs.rms_dev_B_iso (*float*)
The root-mean-square deviation in equivalent isotropic displacement parameters in the domain specified by **_refine_ls_restr_ncs.dom_id** and in the domains against which it was restrained.
The permitted range is $[0.0, \infty)$. **[refine_ls_restr_ncs]**

_refine_ls_restr_ncs.rms_dev_position (*float*)
The root-mean-square deviation in equivalent atom positions in the domain specified by **_refine_ls_restr_ncs.dom_id** and in the domains against which it was restrained.
The permitted range is $[0.0, \infty)$. **[refine_ls_restr_ncs]**

_refine_ls_restr_ncs.weight_B_iso (*float*)
The value of the weighting coefficient used in noncrystallographic symmetry restraint of isotropic displacement parameters in the domain specified by **_refine_ls_restr_ncs.dom_id** to equivalent isotropic displacement parameters in the domains against which it was restrained.

[refine_ls_restr_ncs]

_refine_ls_restr_ncs.weight_position (*float*)
The value of the weighting coefficient used in noncrystallographic symmetry restraint of atom positions in the domain specified by **_refine_ls_restr_ncs.dom_id** to equivalent atom positions in the domains against which it was restrained.

[refine_ls_restr_ncs]

REFINE_LS_RESTR_TYPE

Data items in the REFINE_LS_RESTR_TYPE category record details about the restraint types used in the least-squares refinement.

Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine_ls_restr_type.type**

Example 1 – based on RESTRAIN refinement for the CCP4 test data set toxd.

```
loop_
_refine_ls_restr.type
_refine_ls_restr.number
_refine_ls_restr.dev_ideal
_refine_ls_restr.dev_ideal_target
 'RESTRAIN_Distances < 2.12'              509     0.005 0.022
 'RESTRAIN_Distances 2.12 < D < 2.625'    671     0.016 0.037
 'RESTRAIN_Distances > 2.625'              39     0.034 0.043
 'RESTRAIN_Peptide Planes'                 59     0.002 0.010
 'RESTRAIN_Ring and other planes'          26     0.014 0.010
 'RESTRAIN_rms diffs for Uiso atoms at dist 1.2-1.4'
     212     0.106   .
 'RESTRAIN_rms diffs for Uiso atoms at dist 1.4-1.6'
     288     0.101   .
 'RESTRAIN_rms diffs for Uiso atoms at dist 1.8-2.0'
       6     0.077   .
 'RESTRAIN_rms diffs for Uiso atoms at dist 2.0-2.2'
      10     0.114   .
 'RESTRAIN_rms diffs for Uiso atoms at dist 2.2-2.4'
     215     0.119   .
 'RESTRAIN_rms diffs for Uiso atoms at dist >2.4'
     461     0.106   .

loop_
_refine_ls_restr_type.type
_refine_ls_restr_type.distance_cutoff_low
_refine_ls_restr_type.distance_cutoff_high
 'RESTRAIN_Distances < 2.12'                    .     2.12
 'RESTRAIN_Distances 2.12 < D < 2.625'          2.12  2.625
 'RESTRAIN_Distances > 2.625'                   2.625 .
 'RESTRAIN_Peptide Planes'                      .     .
 'RESTRAIN_Ring and other planes'               .     .
 'RESTRAIN_rms diffs for Uiso atoms at dist 1.2-1.4'
     1.2   1.4
 'RESTRAIN_rms diffs for Uiso atoms at dist 1.4-1.6'
     1.4   1.6
 'RESTRAIN_rms diffs for Uiso atoms at dist 1.8-2.0'
     1.8   2.0
 'RESTRAIN_rms diffs for Uiso atoms at dist 2.0-2.2'
     2.0   2.2
 'RESTRAIN_rms diffs for Uiso atoms at dist 2.2-2.4'
     2.2   2.4
 'RESTRAIN_rms diffs for Uiso atoms at dist >2.4'
     2.4   .
```

_refine_ls_restr_type.distance_cutoff_high (*float*)
The upper limit in ångströms of the distance range applied to the current restraint type.
The permitted range is $[0.0, \infty)$. **[refine_ls_restr_type]**

_refine_ls_restr_type.distance_cutoff_low (*float*)
The lower limit in ångströms of the distance range applied to the current restraint type.
The permitted range is $[0.0, \infty)$. **[refine_ls_restr_type]**

* **_refine_ls_restr_type.type** (*line*)
This data item is a pointer to **_refine_ls_restr.type** in the REFINE_LS_RESTR category.

[refine_ls_restr_type]

REFINE_LS_SHELL

Data items in the REFINE_LS_SHELL category record details about the results of the least-squares refinement broken down into shells of resolution.

Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine_ls_shell.d_res_low**
_refine_ls_shell.d_res_high

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_refine_ls_shell.d_res_low
_refine_ls_shell.d_res_high
_refine_ls_shell.number_reflns_obs
_refine_ls_shell.R_factor_obs
  8.00   4.51   1226   0.196
  4.51   3.48   1679   0.146
  3.48   2.94   2014   0.160
  2.94   2.59   2147   0.182
  2.59   2.34   2127   0.193
  2.34   2.15   2061   0.203
  2.15   2.00   1647   0.188
```

* **_refine_ls_shell.d_res_high** (*float*)
The lowest value for the interplanar spacings for the reflection data in this shell. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[refine_ls_shell]**

* **_refine_ls_shell.d_res_low** (*float*)
The highest value for the interplanar spacings for the reflection data in this shell. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[refine_ls_shell]**

_refine_ls_shell.number_reflns_all (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low**.
The permitted range is $[0, \infty)$. **[refine_ls_shell]**

_refine_ls_shell.number_reflns_obs (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation criterion established by **_reflns.observed_criterion**.
The permitted range is $[0, \infty)$. **[refine_ls_shell]**

_refine_ls_shell.number_reflns_R_free (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' *R* factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.
The permitted range is $[0, \infty)$. **[refine_ls_shell]**

_refine_ls_shell.number_reflns_R_work (*int*)
The number of reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' *R* factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.
The permitted range is $[0, \infty)$. **[refine_ls_shell]**

_refine_ls_shell.percent_reflns_obs (*float*)
The number of reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation criterion established by **_reflns.observed_criterion**, expressed as a percentage of the number of geometrically observable reflections that satisfy the resolution limits.
[refine_ls_shell]

_refine_ls_shell.percent_reflns_R_free (*float*)
The number of reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' *R* factor, expressed as a percentage of the number of geometrically observable reflections that satisfy the reflection limits.
[refine_ls_shell]

_refine_ls_shell.R_factor_all (*float*)
Residual factor *R* for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low**.

$$R = \frac{\sum |F_{\text{obs}} - F_{\text{calc}}|}{\sum |F_{\text{obs}}|},$$

where F_{obs} = the observed structure-factor amplitudes, F_{calc} = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine_ls_shell.wR_factor_all** (alternate).
[refine_ls_shell]

_refine_ls_shell.R_factor_obs (*float*)
Residual factor *R* for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation criterion established by **_reflns.observed_criterion**.

$$R = \frac{\sum |F_{\text{obs}} - F_{\text{calc}}|}{\sum |F_{\text{obs}}|},$$

where F_{obs} = the observed structure-factor amplitudes, F_{calc} = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine_ls_shell.wR_factor_obs** (alternate).
[refine_ls_shell]

_refine_ls_shell.R_factor_R_free (*float*)
Residual factor R for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$.

Related items: **_refine_ls_shell.wR_factor_R_free** (alternate), **_refine_ls_shell.R_factor_R_free_error** (associated error).

[refine_ls_shell]

_refine_ls_shell.R_factor_R_free_error (*float*)
The estimated error in **_refine_ls_shell.R_factor_R_free**. The method used to estimate the error is described in the item **_refine.ls_R_factor_R_free_error_details**.

Related item: **_refine_ls_shell.R_factor_R_free** (associated value).

[refine_ls_shell]

_refine_ls_shell.R_factor_R_work (*float*)
Residual factor R for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$.

Related item: **_refine_ls_shell.wR_factor_R_work** (alternate).

[refine_ls_shell]

_refine_ls_shell.redundancy_reflns_all (*float*)
The ratio of the total number of observations of the reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** to the number of crystallographically unique reflections that satisfy the same limits.

[refine_ls_shell]

_refine_ls_shell.redundancy_reflns_obs (*float*)
The ratio of the total number of observations of the reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation criterion established by **_reflns.observed_criterion** to the number of crystallographically unique reflections that satisfy the same limits.

[refine_ls_shell]

_refine_ls_shell.wR_factor_all (*float*)
Weighted residual factor wR for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$.

Related item: **_refine_ls_shell.R_factor_all** (alternate).

[refine_ls_shell]

_refine_ls_shell.wR_factor_obs (*float*)
Weighted residual factor wR for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation criterion established by **_reflns.observed_criterion**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$.

Related item: **_refine_ls_shell.R_factor_obs** (alternate).

[refine_ls_shell]

_refine_ls_shell.wR_factor_R_free (*float*)
Weighted residual factor wR for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the test reflections (*i.e.* were excluded from the refinement) when the refinement included the calculation of a 'free' R factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.

The permitted range is $[0.0, \infty)$.

Related item: **_refine_ls_shell.R_factor_R_free** (alternate).

[refine_ls_shell]

_refine_ls_shell.wR_factor_R_work (*float*)
Weighted residual factor *wR* for reflections that satisfy the resolution limits established by **_refine_ls_shell.d_res_high** and **_refine_ls_shell.d_res_low** and the observation limit established by **_reflns.observed_criterion**, and that were used as the working reflections (*i.e.* were included in the refinement) when the refinement included the calculation of a 'free' *R* factor. Details of how reflections were assigned to the working and test sets are given in **_reflns.R_free_details**.

$$wR = \left(\frac{\sum \left| w|Y_{\rm obs} - Y_{\rm calc}|^2 \right|}{\sum |wY_{\rm obs}^2|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef** and w = the least-squares weight; the sum is taken over the specified reflections.
The permitted range is $[0.0, \infty)$.
Related item: **_refine_ls_shell.R_factor_R_work** (alternate).
[refine_ls_shell]

REFINE_OCCUPANCY

Data items in the REFINE_OCCUPANCY category record details about the treatment of atom occupancies during refinement.
Category group(s): **inclusive_group**
refine_group
Category key(s): **_refine_occupancy.class**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_refine_occupancy.class
_refine_occupancy.treatment
_refine_occupancy.value
_refine_occupancy.details
 'protein'                 fix  1.00  .
 'solvent'                 fix  1.00  .
 'inhibitor orientation 1'  fix  0.65  .
 'inhibitor orientation 2'  fix  0.35
; The inhibitor binds to the enzyme in two alternative
  conformations. The occupancy of each conformation was
  adjusted so as to result in approximately equal mean
  thermal factors for the atoms in each conformation.
;
```

***_refine_occupancy.class** (*text*)
The class of atoms treated similarly for occupancy refinement.
Examples: 'all', 'protein', 'solvent', 'sugar-phosphate backbone'.
[refine_occupancy]

_refine_occupancy.details (*text*)
A description of special aspects of the occupancy refinement for a class of atoms described in **_refine_occupancy.class**.
Example:

```
; The inhibitor binds to the enzyme in two alternative
conformations. The occupancy of each conformation was
adjusted so as to result in approximately equal mean thermal
factors for the atoms in each conformation.
;
```

[refine_occupancy]

_refine_occupancy.treatment (*ucode*)
The treatment of occupancies for a class of atoms described in **_refine_occupancy.class**.
The data value must be one of the following:
fix fixed
ref refined
[refine_occupancy]

_refine_occupancy.value (*float*)
The value of occupancy assigned to a class of atoms defined in **_refine_occupancy.class**. Meaningful only for atoms with fixed occupancy.
The permitted range is [0.0, 1.0].
Examples: '1.0', '0.41'.
[refine_occupancy]

REFLN

Data items in the REFLN category record details about the reflection data used to determine the ATOM_SITE data items. The REFLN data items refer to individual reflections and must be included in looped lists. The REFLNS data items specify the parameters that apply to all reflections. The REFLNS data items are not looped.
Category group(s): **inclusive_group**
refln_group
Category key(s): **_refln.index_h**
_refln.index_k
_refln.index_l

*Example 1 – based on data set fetod of Todres, Yanovsky, Ermekov & Struchkov [Acta Cryst. (1993), C**49**, 1352–1354].*

```
loop_
_refln.index_h
_refln.index_k
_refln.index_l
_refln.F_squared_calc
_refln.F_squared_meas
_refln.F_squared_sigma
_refln.status
   2   0   0       85.57       58.90      1.45 o
   3   0   0    15718.18    15631.06     30.40 o
   4   0   0    55613.11    49840.09     61.86 o
   5   0   0      246.85      241.86     10.02 o
   6   0   0       82.16       69.97      1.93 o
   7   0   0     1133.62      947.79     11.78 o
   8   0   0     2558.04     2453.33     20.44 o
   9   0   0      283.88      393.66      7.79 o
  10   0   0      283.70      171.98      4.26 o
```

_refln.A_calc (*float*)
_refln_A_calc (*cif_core.dic 2.0.1*)
The calculated value of structure-factor component *A* in electrons.

$$A = |F| \cos(\text{phase}).$$

Related item: **_refln.A_calc_au** (conversion arbitrary). **[refln]**

_refln.A_calc_au (*float*)
The calculated value of structure-factor component *A* in arbitrary units.

$$A = |F| \cos(\text{phase}).$$

Related item: **_refln.A_calc** (conversion arbitrary). **[refln]**

_refln.A_meas (*float*)
_refln_A_meas (*cif_core.dic 2.0.1*)
The measured value of structure-factor component *A* in electrons.

$$A = |F| \cos(\text{phase}).$$

Related item: **_refln.A_meas_au** (conversion arbitrary). **[refln]**

_refln.A_meas_au (*float*)
The measured value of structure-factor component *A* in arbitrary units.

$$A = |F| \cos(\text{phase}).$$

Related item: **_refln.A_meas** (conversion arbitrary). **[refln]**

_refln.B_calc (*float*)
_refln_B_calc (*cif_core.dic 2.0.1*)
The calculated value of structure-factor component B in electrons.

$$B = |F| \sin(\text{phase}).$$

Related item: **_refln.B_calc_au** (conversion arbitrary). **[refln]**

_refln.B_calc_au (*float*)
The calculated value of structure-factor component B in arbitrary units.

$$B = |F| \sin(\text{phase}).$$

Related item: **_refln.B_calc** (conversion arbitrary). **[refln]**

_refln.B_meas (*float*)
_refln_B_meas (*cif_core.dic 2.0.1*)
The measured value of structure-factor component B in electrons.

$$B = |F| \sin(\text{phase}).$$

Related item: **_refln.B_meas_au** (conversion arbitrary). **[refln]**

_refln.B_meas_au (*float*)
The measured value of structure-factor component B in arbitrary units.

$$B = |F| \sin(\text{phase}).$$

Related item: **_refln.B_meas** (conversion arbitrary). **[refln]**

_refln.class_code (*code*)
_refln_class_code (*cif_core.dic 2.3*)
The code identifying the class to which this reflection has been assigned. This code must match a value of **_reflns_class.code**. Reflections may be grouped into classes for a variety of purposes. For example, for modulated structures each reflection class may be defined by the number $m = \sum |m_i|$, where the m_i are the integer coefficients that, in addition to h, k, l, index the corresponding diffraction vector in the basis defined for the reciprocal lattice.
[refln]

*_refln.crystal_id
_refln_crystal_id (*cif_core.dic 2.0.1*)
This data item is a pointer to **_exptl_crystal.id** in the EXPTL_CRYSTAL category.

_refln.d_spacing (*float*)
_refln_d_spacing (*cif_core.dic 2.3*)
The d spacing in ångströms for this reflection. This is related to the $(\sin\theta)/\lambda$ value by the expression **_refln.d_spacing** = 2/(**_refln.sint/lambda**).
The permitted range is $[0.0, \infty)$. **[refln]**

_refln.F_calc (*float*)
_refln_F_calc (*cif_core.dic 2.0.1*)
The calculated value of the structure factor in electrons.
Related item: **_refln.F_calc_au** (conversion arbitrary). **[refln]**

_refln.F_calc_au (*float*)
The calculated value of the structure factor in arbitrary units.
Related item: **_refln.F_calc** (conversion arbitrary). **[refln]**

_refln.F_meas (*float, su*)
_refln_F_meas (*cif_core.dic 2.0.1*)
The measured value of the structure factor in electrons.
Related items: **_refln.F_meas_sigma** (associated esd), **_refln.F_meas_au** (conversion arbitrary). **[refln]**

_refln.F_meas_au (*float, su*)
The measured value of the structure factor in arbitrary units.
Related items: **_refln.F_meas_sigma_au** (associated esd), **_refln.F_meas** (conversion arbitrary). **[refln]**

_refln.F_meas_sigma (*float*)
_refln_F_sigma (*cif_core.dic 2.0.1*)
The standard uncertainty (estimated standard deviation) of **_refln.F_meas** in electrons.
Related items: **_refln.F_meas** (associated value), **_refln.F_meas_sigma_au** (conversion arbitrary). **[refln]**

_refln.F_meas_sigma_au (*float*)
The standard uncertainty (estimated standard deviation) of **_refln.F_meas_au** in arbitrary units.
Related items: **_refln.F_meas_au** (associated value), **_refln.F_meas_sigma** (conversion arbitrary). **[refln]**

_refln.F_squared_calc (*float*)
_refln_F_squared_calc (*cif_core.dic 2.0.1*)
The calculated value of the squared structure factor in electrons squared.
[refln]

_refln.F_squared_meas (*float*)
_refln_F_squared_meas (*cif_core.dic 2.0.1*)
The measured value of the squared structure factor in electrons squared.
[refln]

_refln.F_squared_sigma (*float*)
_refln_F_squared_sigma (*cif_core.dic 2.0.1*)
The standard uncertainty (derived from measurement) of the squared structure factor in electrons squared.
[refln]

_refln.fom (*float*)
The figure of merit m for this reflection.

$$m = \frac{\int P_\alpha \exp(i\alpha)\, \mathrm{d}\alpha}{\int P_\alpha\, \mathrm{d}\alpha},$$

where P_α = the probability that the phase angle α is correct; the integral is taken over the range $\alpha = 0$ to 2π.
The permitted range is $[0.0, \infty)$. **[refln]**

_refln.include_status (*code*)
_refln_include_status (*cif_core.dic 2.3*)
Classification of a reflection so as to indicate its status with respect to inclusion in the refinement and the calculation of R factors.
Related item: **_refln.status** (alternate).
The data value must be one of the following:

o	(lower-case letter o for 'observed') satisfies **_refine.ls_d_res_high**, satisfies **_refine.ls_d_res_low** and exceeds **_reflns.threshold_expression**
<	satisfies **_refine.ls_d_res_high**, satisfies **_refine.ls_d_res_low** and does not exceed **_reflns.threshold_expression**
-	systematically absent reflection
x	unreliable measurement – not used
h	does not satisfy **_refine.ls_d_res_high**
l	does not satisfy **_refine.ls_d_res_low**

[refln]

* **_refln.index_h** (*int*)
_refln_index_h (*cif_core.dic 2.0.1*)
Miller index h of the reflection. The values of the Miller indices in the REFLN category must correspond to the cell defined by cell lengths and cell angles in the CELL category.
[refln]

* **_refln.index_k** (*int*)
_refln_index_k (*cif_core.dic 2.0.1*)
Miller index k of the reflection. The values of the Miller indices in the REFLN category must correspond to the cell defined by cell lengths and cell angles in the CELL category.
[refln]

* **_refln.index_l** (*int*)
_refln_index_l (*cif_core.dic 2.0.1*)
Miller index l of the reflection. The values of the Miller indices in the REFLN category must correspond to the cell defined by cell lengths and cell angles in the CELL category.
[refln]

_refln.intensity_calc (*float*)
_refln_intensity_calc (*cif_core.dic 2.0.1*)
The calculated value of the intensity in the same units as **_refln.intensity_meas**.
[refln]

_refln.intensity_meas (*float*)
_refln_intensity_meas (*cif_core.dic 2.0.1*)
The measured value of the intensity.
[refln]

_refln.intensity_sigma (*float*)
_refln_intensity_sigma (*cif_core.dic 2.0.1*)
The standard uncertainty (derived from measurement) of the intensity in the same units as **_refln.intensity_meas**.
[refln]

_refln.mean_path_length_tbar (*float*)
_refln_mean_path_length_tbar (*cif_core.dic 2.3*)
Mean path length in millimetres through the crystal for this reflection.
The permitted range is $[0.0, \infty)$. **[refln]**

_refln.phase_calc (*float*)
_refln_phase_calc (*cif_core.dic 2.0.1*)
The calculated structure-factor phase in degrees.
[refln]

_refln.phase_meas (*float*)
_refln_phase_meas (*cif_core.dic 2.0.1*)
The measured structure-factor phase in degrees.
[refln]

_refln.refinement_status (*ucode*)
_refln_refinement_status (*cif_core.dic 2.0.1*)
Status of a reflection in the structure-refinement process.
The data value must be one of the following:

`incl`	included in least-squares process
`excl`	excluded from least-squares process
`extn`	excluded due to extinction

Where no value is given, the assumed value is '`incl`'. **[refln]**

* **_refln.scale_group_code**
_refln_scale_group_code (*cif_core.dic 2.0.1*)
This data item is a pointer to **_reflns_scale.group_code** in the REFLNS_SCALE category.

_refln.sint_over_lambda (*float*)
_refln_sint/lambda (*cif_core.dic 2.0.1*)
The $(\sin\theta)/\lambda$ value in reciprocal ångströms for this reflection.
The permitted range is $[0.0, \infty)$. **[refln]**

_refln.status (*ucode*)
_refln_observed_status (*cif_core.dic 2.0.1*)
Classification of a reflection so as to indicate its status with respect to inclusion in the refinement and the calculation of R factors.
The data value must be one of the following:

`o`	satisfies **_refine.ls_d_res_high**, satisfies **_refine.ls_d_res_low**, observed by **_reflns.observed_criterion**, not flagged as systematically absent, not flagged as unreliable
`<`	satisfies **_refine.ls_d_res_high**, satisfies **_refine.ls_d_res_low**, unobserved by **_reflns.observed_criterion**, not flagged as systematically absent, not flagged as unreliable
`-`	systematically absent reflection
`x`	unreliable measurement – not used
`h`	does not satisfy **_refine.ls_d_res_high**
`l`	does not satisfy **_refine.ls_d_res_low**
`f`	satisfies **_refine.ls_d_res_high**, satisfies **_refine.ls_d_res_low**, observed by **_reflns.observed_criterion**, not flagged as systematically absent, not flagged as unreliable, excluded from refinement so as to be included in the calculation of a 'free' R factor

[refln]

_refln.symmetry_epsilon (*int*)
_refln_symmetry_epsilon (*cif_core.dic 2.0.1*)
The symmetry reinforcement factor corresponding to the number of times the reflection indices are generated identically from the space-group symmetry operations.
The permitted range is $[1, 48]$. **[refln]**

_refln.symmetry_multiplicity (*int*)
_refln_symmetry_multiplicity (*cif_core.dic 2.0.1*)
The number of symmetry-equivalent reflections. The equivalent reflections have the same structure-factor magnitudes because of the space-group symmetry and the Friedel relationship.
The permitted range is $[1, 48]$. **[refln]**

_refln.wavelength (*float*)
_refln_wavelength (*cif_core.dic 2.0.1*)
The mean wavelength in ångströms of radiation used to measure this reflection. This is an important parameter for data collected using energy-dispersive detectors or the Laue method.
The permitted range is $[0.0, \infty)$. **[refln]**

* **_refln.wavelength_id**
_refln_wavelength_id (*cif_core.dic 2.0.1*)
This data item is a pointer to **_diffrn_radiation.wavelength_id** in the DIFFRN_RADIATION category.

REFLN_SYS_ABS

Data items in the REFLN_SYS_ABS category record details about the reflection data that should be systematically absent, given the designated space group.

Category group(s): **inclusive_group**
refln_group
Category key(s): **_refln_sys_abs.index_h**
_refln_sys_abs.index_k
_refln_sys_abs.index_l

Example 1 – hypothetical example.

```
loop_
_refln_sys_abs.index_h
_refln_sys_abs.index_k
_refln_sys_abs.index_l
_refln_sys_abs.I
_refln_sys_abs.sigmaI
_refln_sys_abs.I_over_sigmaI
 0  3  0   28.32  22.95  1.23
 0  5  0   14.11  16.38  0.86
 0  7  0  114.81  20.22  5.67
 0  9  0   32.99  24.51  1.35
```

_refln_sys_abs.I (*float, su*)

_ebi_refln_sys_abs.I (*ebi_extensions 1.0*)

The measured value of the intensity in arbitrary units.

Related item: **_refln_sys_abs.sigmaI** (associated esd). **[refln_sys_abs]**

_refln_sys_abs.I_over_sigmaI (*float*)

_ebi_refln_sys_abs.I_over_sigma (*ebi_extensions 1.0*)

The ratio of **_refln_sys_abs.I** to **_refln_sys_abs.sigmaI**. Used to evaluate whether a reflection that should be systematically absent according to the designated space group is in fact absent.

[refln_sys_abs]

*_refln_sys_abs.index_h (*int*)

_ebi_refln_sys_abs.h (*ebi_extensions 1.0*)

Miller index *h* of the reflection. The values of the Miller indices in the REFLN_SYS_ABS category must correspond to the cell defined by cell lengths and cell angles in the CELL category.

[refln_sys_abs]

*_refln_sys_abs.index_k (*int*)

_ebi_refln_sys_abs.k (*ebi_extensions 1.0*)

Miller index *k* of the reflection. The values of the Miller indices in the REFLN_SYS_ABS category must correspond to the cell defined by cell lengths and cell angles in the CELL category.

[refln_sys_abs]

*_refln_sys_abs.index_l (*int*)

_ebi_refln_sys_abs.l (*ebi_extensions 1.0*)

Miller index *l* of the reflection. The values of the Miller indices in the REFLN_SYS_ABS category must correspond to the cell defined by cell lengths and cell angles in the CELL category.

[refln_sys_abs]

_refln_sys_abs.sigmaI (*float*)

_ebi_refln_sys_abs.sigmaI (*ebi_extensions 1.0*)

The standard uncertainty (estimated standard deviation) of **_refln_sys_abs.I** in arbitrary units.

Related item: **_refln_sys_abs.I** (associated value). **[refln_sys_abs]**

REFLNS

Data items in the REFLNS category record details about the reflection data used to determine the ATOM_SITE data items. The REFLN data items refer to individual reflections and must be included in looped lists. The REFLNS data items specify the parameters that apply to all reflections. The REFLNS data items are not looped.

Category group(s): **inclusive_group**
refln_group
Category key(s): **_reflns.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_reflns.entry_id                      '5HVP'
_reflns.data_reduction_method
; Xengen program scalei. Anomalous pairs were merged. Scaling
  proceeded in several passes, beginning with 1-parameter
  fit and ending with 3-parameter fit.
;
_reflns.data_reduction_details
; Merging and scaling based on only those reflections
  with I > \s(I).
;

_reflns.d_resolution_high             2.00
_reflns.d_resolution_low              8.00

_reflns.limit_h_max                   22
_reflns.limit_h_min                   0
_reflns.limit_k_max                   46
_reflns.limit_k_min                   0
_reflns.limit_l_max                   57
_reflns.limit_l_min                   0

_reflns.number_obs                    7228
_reflns.observed_criterion            '> 1 \s(I)'
_reflns.details                       none
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_reflns.limit_h_min              0
_reflns.limit_h_max              6
_reflns.limit_k_min              0
_reflns.limit_k_max              17
_reflns.limit_l_min              0
_reflns.limit_l_max              22
_reflns.number_all               1592
_reflns.number_obs               1408
_reflns.observed_criterion       F_>_6.0_\s(F)
_reflns.d_resolution_high        0.8733
_reflns.d_resolution_low         11.9202
```

_reflns.B_iso_Wilson_estimate (*float*)

The value of the overall isotropic displacement parameter estimated from the slope of the Wilson plot.

[reflns]

_reflns.d_resolution_high (*float*)

_reflns_d_resolution_high (*cif_core.dic 2.0.1*)

The smallest value for the interplanar spacings for the reflection data. This is called the highest resolution.

The permitted range is $[0.0, \infty)$. **[reflns]**

_reflns.d_resolution_low (*float*)

_reflns_d_resolution_low (*cif_core.dic 2.0.1*)

The largest value for the interplanar spacings for the reflection data. This is called the lowest resolution.

The permitted range is $[0.0, \infty)$. **[reflns]**

_reflns.data_reduction_details (*text*)

A description of special aspects of the data-reduction procedures.

Example:

```
; Merging and scaling based on only those
  reflections with I > sig(I).
;
```

[reflns]

_reflns.data_reduction_method (*text*)

The method used for data reduction. Note that this is not the computer program used, which is described in the SOFTWARE category, but the method itself. This data item should be used to describe significant methodological options used within the data-reduction programs.

Example:

```
; Profile fitting by method of Kabsch (1987).
  Scaling used spherical harmonic coefficients.
;
```

[reflns]

_reflns.details (*text*)

_reflns_special_details (*cif_core.dic 2.0.1*)

A description of reflection data not covered by other data names. This should include details of the Friedel pairs.

[reflns]

***_reflns.entry_id**

This data item is a pointer to **_entry.id** in the ENTRY category.

_reflns.Friedel_coverage (*float*)

_reflns_Friedel_coverage (*cif_core.dic 2.3*)

The proportion of Friedel-related reflections present in the number of 'independent' reflections specified by the item **_reflns.number_all**. This proportion is calculated as the ratio

$$\frac{[N(\text{crystal class}) - N(\text{Laue symmetry})]}{N(\text{Laue symmetry})},$$

where, working from the DIFFRN_REFLN list, N(Crystal class) is the number of reflections obtained on averaging under the symmetry of the crystal class and N(Laue symmetry) is the number of reflections obtained on averaging under the Laue symmetry.

Examples: (*a*) For centrosymmetric structures, the value of **_reflns.Friedel_coverage** is necessarily equal to 0.0 as the crystal class is identical to the Laue symmetry. (*b*) For whole-sphere data for a crystal in the space group $P1$, **_reflns.Friedel_coverage** is equal to 1.0, as no reflection hkl is equivalent to $-h-k-l$ in the crystal class and all Friedel pairs $\{hkl; -h-k-l\}$ have been measured. (*c*) For whole-sphere data in space group *Pmm*2, **_reflns.Friedel_coverage** will be < 1.0 because although reflections hkl and $-h-k-l$ are not equivalent when hkl indices are nonzero, they are when $l = 0$. (*d*) For a crystal in space group *Pmm*2, measurements of the two inequivalent octants $h \geq 0, k \geq 0, l$ lead to the same value as in (*c*), whereas measurements of the two equivalent octants $h \geq 0, k, l \geq 0$ will lead to a value of zero for **_reflns.Friedel_coverage**.

The permitted range is [0.0, 1.0]. **[reflns]**

_reflns.limit_h_max (*int*)

_reflns_limit_h_max (*cif_core.dic 2.0.1*)

Maximum value of the Miller index h for the reflection data. This need not have the same value as **_diffrn_reflns.limit_h_max**.

[reflns]

_reflns.limit_h_min (*int*)

_reflns_limit_h_min (*cif_core.dic 2.0.1*)

Minimum value of the Miller index h for the reflection data. This need not have the same value as **_diffrn_reflns.limit_h_min**.

[reflns]

_reflns.limit_k_max (*int*)

_reflns_limit_k_max (*cif_core.dic 2.0.1*)

Maximum value of the Miller index k for the reflection data. This need not have the same value as **_diffrn_reflns.limit_k_max**.

[reflns]

_reflns.limit_k_min (*int*)

_reflns_limit_k_min (*cif_core.dic 2.0.1*)

Minimum value of the Miller index k for the reflection data. This need not have the same value as **_diffrn_reflns.limit_k_min**.

[reflns]

_reflns.limit_l_max (*int*)

_reflns_limit_l_max (*cif_core.dic 2.0.1*)

Maximum value of the Miller index l for the reflection data. This need not have the same value as **_diffrn_reflns.limit_l_max**.

[reflns]

_reflns.limit_l_min (*int*)

_reflns_limit_l_min (*cif_core.dic 2.0.1*)

Minimum value of the Miller index l for the reflection data. This need not have the same value as **_diffrn_reflns.limit_l_min**.

[reflns]

_reflns.number_all (*int*)

_reflns_number_total (*cif_core.dic 2.0.1*)

The total number of reflections in the REFLN list (not the DIFFRN_REFLN list). This number may contain Friedel-equivalent reflections according to the nature of the structure and the procedures used. The item **_reflns.details** describes the reflection data.

The permitted range is $[0, \infty)$. **[reflns]**

_reflns.number_gt (*int*)

_reflns_number_gt (*cif_core.dic 2.3*)

The number of reflections in the REFLN list (not the DIFFRN_REFLN list) that are significantly intense, satisfying the criterion specified by **_reflns.threshold_expression**. This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Any special characteristics of the reflections included in the REFLN list should be described using the item **_reflns.details**.

The permitted range is $[0, \infty)$. **[reflns]**

_reflns.number_obs (*int*)

_reflns_number_observed (*cif_core.dic 2.0.1*)

The number of reflections in the REFLN list (not the DIFFRN_REFLN list) classified as observed (see **_reflns.observed_criterion**). This number may contain Friedel-equivalent reflections according to the nature of the structure and the procedures used.

The permitted range is $[0, \infty)$. **[reflns]**

_reflns.observed_criterion (*text*)

_reflns_observed_criterion (*cif_core.dic 2.0.1*)

The criterion used to classify a reflection as 'observed'. This criterion is usually expressed in terms of a $\sigma(I)$ or $\sigma(F)$ threshold.

Related items: **_reflns.observed_criterion_sigma_F** (alternate), **_reflns.observed_criterion_sigma_I** (alternate), **_reflns.observed_criterion_I_min** (alternate), **_reflns.observed_criterion_I_max** (alternate), **_reflns.observed_criterion_F_min** (alternate), **_reflns.observed_criterion_F_max** (alternate) .

Example: '**>2sigma(I)**'. **[reflns]**

_reflns.observed_criterion_F_max (*float*)

The criterion used to classify a reflection as 'observed' expressed as an upper limit for the value of F.

Related items: **_reflns.observed_criterion** (alternate), **_reflns.observed_criterion_I_max** (convention). **[reflns]**

_reflns.observed_criterion_F_min (*float*)
The criterion used to classify a reflection as 'observed' expressed as a lower limit for the value of F.
Related items: **_reflns.observed_criterion** (alternate), **_reflns.observed_criterion_I_min** (convention). **[reflns]**

_reflns.observed_criterion_I_max (*float*)
The criterion used to classify a reflection as 'observed' expressed as an upper limit for the value of I.
Related items: **_reflns.observed_criterion** (alternate), **_reflns.observed_criterion_F_max** (convention). **[reflns]**

_reflns.observed_criterion_I_min (*float*)
The criterion used to classify a reflection as 'observed' expressed as a lower limit for the value of I.
Related items: **_reflns.observed_criterion** (alternate), **_reflns.observed_criterion_F_min** (convention). **[reflns]**

_reflns.observed_criterion_sigma_F (*float*)
The criterion used to classify a reflection as 'observed' expressed as a multiple of the value of $\sigma(F)$.
Related items: **_reflns.observed_criterion** (alternate), **_reflns.observed_criterion_sigma_I** (convention). **[reflns]**

_reflns.observed_criterion_sigma_I (*float*)
The criterion used to classify a reflection as 'observed' expressed as a multiple of the value of $\sigma(I)$.
Related items: **_reflns.observed_criterion** (alternate), **_reflns.observed_criterion_sigma_F** (convention). **[reflns]**

_reflns.percent_possible_obs (*float*)
The percentage of geometrically possible reflections represented by reflections that satisfy the resolution limits established by **_reflns.d_resolution_high** and **_reflns.d_resolution_low** and the observation limit established by **_reflns.observed_criterion**.
The permitted range is $[0.0, \infty)$. **[reflns]**

_reflns.R_free_details (*text*)
A description of the method by which a subset of reflections was selected for exclusion from refinement so as to be used in the calculation of a 'free' R factor.
Example:
```
; The data set was sorted with l varying most rapidly and h
varying least rapidly. Every 10th reflection in this sorted
list was excluded from refinement and included in the
calculation of a 'free' R factor.
;
```
[reflns]

_reflns.Rmerge_F_all (*float*)
Residual factor $R_{\rm merge}$ for all reflections that satisfy the resolution limits established by **_reflns.d_resolution_high** and **_reflns.d_resolution_low**.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F\rangle|)}{\sum_i(\sum_j \langle F\rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F\rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$. **[reflns]**

_reflns.Rmerge_F_obs (*float*)
Residual factor $R_{\rm merge}$ for reflections that satisfy the resolution limits established by **_reflns.d_resolution_high** and **_reflns.d_resolution_low** and the observation limit established by **_reflns.observed_criterion**.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F\rangle|)}{\sum_i(\sum_j \langle F\rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F\rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$. **[reflns]**

_reflns.threshold_expression (*text*)
_reflns_threshold_expression (*cif_core.dic 2.3*)
The threshold, usually based on multiples of $u(I)$, $u(F^2)$ or $u(F)$, that serves to identify significantly intense reflections, the number of which is given by **_reflns.number_gt**. These reflections are used in the calculation of **_refine.ls_R_factor_gt**.
Related item: **_reflns.observed_criterion** (alternate) .
Example: 'I>2u(I)'. **[reflns]**

REFLNS_CLASS

Data items in the REFLNS_CLASS category record details of the reflections used to determine the structural parameters for each reflection class.
Category key(s): **_reflns_class.code**

Example 1 – example corresponding to the one-dimensional incommensurately modulated structure of K_2SeO_4.

```
loop_
 _reflns_class.number_gt
 _reflns_class.code
     584     'Main'
     226     'Sat1'
     50      'Sat2'
```

***_reflns_class.code** (*code*)
_reflns_class_code (*cif_core.dic 2.3*)
The code identifying a certain reflection class.
Examples: '1', 'm1', 's2'. **[reflns_class]**

_reflns_class.d_res_high (*float*)
_reflns_class_d_res_high (*cif_core.dic 2.3*)
For each reflection class, the smallest value in ångströms for the interplanar spacings for the reflections used in the refinement. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

_reflns_class.d_res_low (*float*)
_reflns_class_d_res_low (*cif_core.dic 2.3*)
For each reflection class, the largest value in ångströms for the interplanar spacings for the reflections used in the refinement. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

_reflns_class.description (*text*)
_reflns_class_description (*cif_core.dic 2.3*)
Description of each reflection class.
Examples: 'm=1 first order satellites', 'H0L0 common projection reflections'. **[reflns_class]**

_reflns_class.number_gt (*int*)
_reflns_class_number_gt (*cif_core.dic 2.3*)
For each reflection class, the number of significantly intense reflections (see **_reflns.threshold_expression**) in the REFLN list (not the DIFFRN_REFLN list). This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Any special characteristics of the reflections included in the REFLN list should be described using the item **_reflns.details**.
The permitted range is $[0, \infty)$. **[reflns_class]**

_reflns_class.number_total (*int*)
_reflns_class_number_total (*cif_core.dic 2.3*)
For each reflection class, the total number of reflections in the REFLN list (not the DIFFRN_REFLN list). This may include Friedel-equivalent reflections (*i.e.* those which are symmetry-equivalent under the Laue symmetry but inequivalent under the crystal class) according to the nature of the structure and the procedures used. Any special characteristics of the reflections included in the REFLN list should be described using the item **_reflns.details**.
The permitted range is $[0, \infty)$. **[reflns_class]**

_reflns_class.R_factor_all (*float*)
_reflns_class_R_factor_all (*cif_core.dic 2.3*)
For each reflection class, the residual factor for all reflections included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class.d_res_high** and **_reflns_class.d_res_low**. This is the conventional *R* factor. See also the definition of **_reflns_class.wR_factor_all**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

_reflns_class.R_factor_gt (*float*)
_reflns_class_R_factor_gt (*cif_core.dic 2.3*)
For each reflection class, the residual factor for significantly intense reflections (see **_reflns.threshold_expression**) included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class.d_res_high** and **_reflns_class.d_res_low**. This is the conventional *R* factor. See also the definition of **_reflns_class.wR_factor_all**.

$$R = \frac{\sum |F_{\rm obs} - F_{\rm calc}|}{\sum |F_{\rm obs}|},$$

where $F_{\rm obs}$ = the observed structure-factor amplitudes, $F_{\rm calc}$ = the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

_reflns_class.R_Fsqd_factor (*float*)
_reflns_class_R_Fsqd_factor (*cif_core.dic 2.3*)
For each reflection class, the residual factor $R(F^2)$ calculated on the squared amplitudes of the observed and calculated structure factors for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns.threshold_expression**) and included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class.d_res_high** and **_reflns_class.d_res_low**.

$$R(F^2) = \frac{\sum |F^2_{\rm obs} - F^2_{\rm calc}|}{\sum |F^2_{\rm obs}|},$$

where $F^2_{\rm obs}$ = squares of the observed structure-factor amplitudes, $F^2_{\rm calc}$ = squares of the calculated structure-factor amplitudes and the sum is taken over the reflections of this class.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

_reflns_class.R_I_factor (*float*)
_reflns_class_R_I_factor (*cif_core.dic 2.3*)
For each reflection class, the residual factor $R(I)$ for the reflections judged significantly intense (*i.e.* satisfying the threshold specified by **_reflns.threshold_expression**) and included in the refinement. This is most often calculated in Rietveld refinements against powder data, where it is referred to as R_B or $R_{\rm Bragg}$.

$$R(I) = \frac{\sum |I_{\rm obs} - I_{\rm calc}|}{\sum |I_{\rm obs}|},$$

where $I_{\rm obs}$ = the net observed intensities, $I_{\rm calc}$ = the net calculated intensities and the sum is taken over the reflections of this class.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

_reflns_class.wR_factor_all (*float*)
_reflns_class_wR_factor_all (*cif_core.dic 2.3*)
For each reflection class, the weighted residual factors for all reflections included in the refinement. The reflections also satisfy the resolution limits established by **_reflns_class.d_res_high** and **_reflns.class_d_res_low**. See also **_reflns_class.R_factor_** definitions.

$$wR = \left(\frac{\sum |w|Y_{\rm obs} - Y_{\rm calc}|^2|}{\sum |wY^2_{\rm obs}|} \right)^{1/2},$$

where $Y_{\rm obs}$ = the observed amplitude specified by **_refine.ls_structure_factor_coef**, $Y_{\rm calc}$ = the calculated amplitude specified by **_refine.ls_structure_factor_coef**, w = the least-squares weight and the sum is taken over the reflections of this class.
The permitted range is $[0.0, \infty)$. **[reflns_class]**

REFLNS_SCALE

Data items in the REFLNS_SCALE category record details about the structure-factor scales. They are referenced from within the REFLN list through **_refln.scale_group_code**.
Category group(s): **inclusive_group**
refln_group
Category key(s): **_reflns_scale.group_code**

Example 1 – based on laboratory records for the collagen-like peptide [(POG)$_4$ EKG (POG)$_5$]$_3$.

```
_reflns_scale.group_code    SG1
_reflns_scale.meas_F        4.0
```

***_reflns_scale.group_code** (*line*)
_reflns_scale_group_code (*cif_core.dic 2.0.1*)
The code identifying a scale **_reflns_scale.meas_F**, **_reflns_scale.meas_F_squared** or **_reflns_scale.meas_intensity**. These are linked to the REFLN list by the **_refln.scale_group_code**. These codes need not correspond to those in the DIFFRN_SCALE list.
The following item(s) have an equivalent role in their respective categories:
_refln.scale_group_code.
Examples: '1', '2', 'c1', 'c2'. **[reflns_scale]**

_reflns_scale.meas_F (*float*)
_reflns_scale_meas_F (*cif_core.dic 2.0.1*)
A scale associated with **_reflns_scale.group_code**.
The permitted range is $[0.0, \infty)$. **[reflns_scale]**

_reflns_scale.meas_F_squared (*float*)
_reflns_scale_meas_F_squared (*cif_core.dic 2.0.1*)
A scale associated with **_reflns_scale.group_code**.
The permitted range is $[0.0, \infty)$. **[reflns_scale]**

_reflns_scale.meas_intensity (*float*)
_reflns_scale_meas_intensity (*cif_core.dic 2.0.1*)
A scale associated with **_reflns_scale.group_code**.
The permitted range is $[0.0, \infty)$. **[reflns_scale]**

REFLNS_SHELL

Data items in the REFLNS_SHELL category record details about the reflection data used to determine the ATOM_SITE data items broken down into shells of resolution.

Category group(s): **inclusive_group**
refln_group
Category key(s): **_reflns_shell.d_res_high**
_reflns_shell.d_res_low

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_reflns_shell.d_res_high
_reflns_shell.d_res_low
_reflns_shell.meanI_over_sigI_obs
_reflns_shell.number_measured_obs
_reflns_shell.number_unique_obs
_reflns_shell.percent_possible_obs
_reflns_shell.Rmerge_F_obs
  31.38  3.82  69.8  9024  2540  96.8   1.98
   3.82  3.03  26.1  7413  2364  95.1   3.85
   3.03  2.65  10.5  5640  2123  86.2   6.37
   2.65  2.41   6.4  4322  1882  76.8   8.01
   2.41  2.23   4.3  3247  1714  70.4   9.86
   2.23  2.10   3.1  1140   812  33.3  13.99
```

***_reflns_shell.d_res_high** (*float*)
_reflns_shell_d_res_high (*cif_core.dic 2.0.1*)
The smallest value in ångströms for the interplanar spacings for the reflections in this shell. This is called the highest resolution.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

***_reflns_shell.d_res_low** (*float*)
_reflns_shell_d_res_low (*cif_core.dic 2.0.1*)
The highest value in ångströms for the interplanar spacings for the reflections in this shell. This is called the lowest resolution.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

_reflns_shell.meanI_over_sigI_all (*float*)
_reflns_shell_meanI_over_sigI_all (*cif_core.dic 2.0.1*)
The ratio of the mean of the intensities of all reflections in this shell to the mean of the standard uncertainties of the intensities of all reflections in this shell.
[reflns_shell]

_reflns_shell.meanI_over_sigI_gt (*float*)
_reflns_shell_meanI_over_sigI_gt (*cif_core.dic 2.3*)
The ratio of the mean of the intensities of the significantly intense reflections (see **_reflns.threshold_expression**) in this shell to the mean of the standard uncertainties of the intensities of the significantly intense reflections in this shell.
Related item: **_reflns_shell.meanI_over_uI_gt** (replaces). **[reflns_shell]**

_reflns_shell.meanI_over_sigI_obs (*float*)
_reflns_shell_meanI_over_sigI_obs (*cif_core.dic 2.0.1*)
The ratio of the mean of the intensities of the reflections classified as 'observed' (see **_reflns.observed_criterion**) in this shell to the mean of the standard uncertainties of the intensities of the 'observed' reflections in this shell.
[reflns_shell]

_reflns_shell.meanI_over_uI_all (*float*)
_reflns_shell_meanI_over_uI_all (*cif_core.dic 2.3*)
The ratio of the mean of the intensities of all reflections in this shell to the mean of the standard uncertainties of the intensities of all reflections in this shell.
Related item: **_reflns_shell.meanI_over_sigI_all** (alternate).
[reflns_shell]

_reflns_shell.meanI_over_uI_gt (*float*)
_reflns_shell_meanI_over_uI_gt (*cif_core.dic 2.3*)
The ratio of the mean of the intensities of the significantly intense reflections (see **_reflns.threshold_expression**) in this shell to the mean of the standard uncertainties of the intensities of the significantly intense reflections in this shell.
Related items: **_reflns_shell.meanI_over_sigI_gt** (alternate), **_reflns_shell.meanI_over_sigI_obs** (alternate). **[reflns_shell]**

_reflns_shell.number_measured_all (*int*)
_reflns_shell_number_measured_all (*cif_core.dic 2.0.1*)
The total number of reflections measured for this shell.
[reflns_shell]

_reflns_shell.number_measured_gt (*int*)
_reflns_shell_number_measured_gt (*cif_core.dic 2.3*)
The number of significantly intense reflections (see **_reflns.threshold_expression**) measured for this shell.
The permitted range is $[0, \infty)$.
Related item: **_reflns_shell.number_measured_obs** (alternate).
[reflns_shell]

_reflns_shell.number_measured_obs (*int*)
_reflns_shell_number_measured_obs (*cif_core.dic 2.0.1*)
The number of reflections classified as 'observed' (see **_reflns.observed_criterion**) for this shell.
[reflns_shell]

_reflns_shell.number_possible (*int*)
_reflns_shell_number_possible (*cif_core.dic 2.0.1*)
The number of unique reflections it is possible to measure in this shell.
The permitted range is $[0, \infty)$. **[reflns_shell]**

_reflns_shell.number_unique_all (*int*)
_reflns_shell_number_unique_all (*cif_core.dic 2.0.1*)
The total number of measured reflections which are symmetry-unique after merging for this shell.
[reflns_shell]

_reflns_shell.number_unique_gt (*int*)
_reflns_shell_number_unique_gt (*cif_core.dic 2.3*)
The total number of significantly intense reflections (see **_reflns.threshold_expression**) resulting from merging measured symmetry-equivalent reflections for this resolution shell.
The permitted range is $[0, \infty)$.
Related item: **_reflns_shell.number_unique_obs** (alternate). **[reflns_shell]**

_reflns_shell.number_unique_obs (*int*)
_reflns_shell_number_unique_obs (*cif_core.dic 2.0.1*)
The total number of measured reflections classified as 'observed' (see **_reflns.observed_criterion**) which are symmetry-unique after merging for this shell.
[reflns_shell]

_reflns_shell.percent_possible_all (*float*)
_reflns_shell_percent_possible_all (*cif_core.dic 2.0.1*)
The percentage of geometrically possible reflections represented by all reflections measured for this shell.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

_reflns_shell.percent_possible_gt (*float*)
_reflns_shell_percent_possible_gt (*cif_core.dic 2.3*)
The percentage of geometrically possible reflections represented by significantly intense reflections (see **_reflns.threshold_expression**) measured for this shell.
The permitted range is [0.0, 100.0].
Related item: **_reflns_shell.percent_possible_obs** (alternate).
[reflns_shell]

_reflns_shell.percent_possible_obs (*float*)
_reflns_shell_percent_possible_obs (*cif_core.dic 2.0.1*)
The percentage of geometrically possible reflections represented by reflections classified as 'observed' (see **_reflns.observed_criterion**) for this shell.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

_reflns_shell.Rmerge_F_all (*float*)
_reflns_shell_Rmerge_F_all (*cif_core.dic 2.0.1*)
Residual factor $R_{\rm merge}$ for all reflections that satisfy the resolution limits established by **_reflns_shell.d_res_high** and **_reflns_shell.d_res_low**.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F \rangle|)}{\sum_i(\sum_j \langle F \rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F \rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

_reflns_shell.Rmerge_F_gt (*float*)
_reflns_shell_Rmerge_F_gt (*cif_core.dic 2.3*)
The value of $R_{\rm merge}(F)$ for significantly intense reflections (see **_reflns.threshold_expression**) in a given shell.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F \rangle|)}{\sum_i(\sum_j \langle F \rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F \rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$.
Related item: **_reflns_shell.Rmerge_F_obs** (alternate). **[reflns_shell]**

_reflns_shell.Rmerge_F_obs (*float*)
_reflns_shell_Rmerge_F_obs (*cif_core.dic 2.0.1*)
Residual factor $R_{\rm merge}$ for reflections that satisfy the resolution limits established by **_reflns_shell.d_res_high** and **_reflns_shell.d_res_low** and the observation criterion established by **_reflns.observed_criterion**.

$$R_{\rm merge} = \frac{\sum_i(\sum_j |F_j - \langle F \rangle|)}{\sum_i(\sum_j \langle F \rangle)},$$

where F_j = the amplitude of the jth observation of reflection i, $\langle F \rangle$ = the mean of the amplitudes of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

_reflns_shell.Rmerge_I_all (*float*)
_reflns_shell_Rmerge_I_all (*cif_core.dic 2.0.1*)
The value of $R_{\rm merge}(I)$ for all reflections in a given shell.

$$R_{\rm merge}(I) = \frac{\sum_i(\sum_j |I_j - \langle I \rangle|)}{\sum_i(\sum_j \langle I \rangle)},$$

where I_j = the intensity of the jth observation of reflection i, $\langle I \rangle$ = the mean of the intensities of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

_reflns_shell.Rmerge_I_gt (*float*)
_reflns_shell_Rmerge_I_gt (*cif_core.dic 2.3*)
The value of $R_{\rm merge}(I)$ for significantly intense reflections (see **_reflns.threshold_expression**) in a given shell.

$$R_{\rm merge}(I) = \frac{\sum_i(\sum_j |I_j - \langle I \rangle|)}{\sum_i(\sum_j \langle I \rangle)},$$

where I_j = the intensity of the jth observation of reflection i, $\langle I \rangle$ = the mean of the intensities of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$.
Related item: **_reflns_shell.Rmerge_I_obs** (alternate). **[reflns_shell]**

_reflns_shell.Rmerge_I_obs (*float*)
_reflns_shell_Rmerge_I_obs (*cif_core.dic 2.0.1*)
The value of $R_{\rm merge}(I)$ for reflections classified as 'observed' (see **_reflns.observed_criterion**) in a given shell.

$$R_{\rm merge}(I) = \frac{\sum_i(\sum_j |I_j - \langle I \rangle|)}{\sum_i(\sum_j \langle I \rangle)},$$

where I_j = the intensity of the jth observation of reflection i, $\langle I \rangle$ = the mean of the intensities of all observations of reflection i, $\sum_i$ is taken over all reflections and $\sum_j$ is taken over all observations of each reflection.
The permitted range is $[0.0, \infty)$. **[reflns_shell]**

SOFTWARE

Data items in the SOFTWARE category record details about the software used in the structure analysis, which implies any software used in the generation of any data items associated with the structure determination and structure representation. These data items allow computer programs to be referenced in more detail than data items in the COMPUTING category do.

Category group(s): **inclusive_group**
computing_group
Category key(s): **_software.name**
_software.version

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_software.name
_software.version
_software.date
_software.type
_software.contact_author
_software.contact_author_email
_software.location
_software.classification
_software.citation_id
_software.language
_software.compiler_name
_software.compiler_version
_software.hardware
_software.os
_software.os_version
_software.dependencies
_software.mods
_software.description
   Prolsq   unknown  .   program   'Wayne A. Hendrickson'  ?
   'ftp://rosebud.sdsc.edu/pub/sdsc/xtal/CCP4/ccp4/'
   refinement ref5  Fortran
   'Convex Fortran' v8.0 'Convex C220' ConvexOS  v10.1
   'Requires that Protin be run first'    optimized
   'restrained least-squares refinement'
```

_software.citation_id
This data item is a pointer to **_citation.id** in the CITATION category.

_software.classification (*uline*)
The classification of the program according to its major function.
Examples: 'data collection', 'data reduction', 'phasing', 'model building', 'refinement', 'validation', 'other'. **[software]**

_software.compiler_name (*line*)
The compiler used to compile the software.
Examples: 'Convex Fortran', 'gcc', 'DEC C'. **[software]**

_software.compiler_version (*line*)
The version of the compiler used to compile the software.
Examples: '3.1', '2.1 alpha'. **[software]**

_software.contact_author (*line*)
The recognized contact author of the software. This could be the original author, someone who has modified the code or someone who maintains the code. It should be the person most commonly associated with the code.
Examples: 'T. Alwyn Jones', 'Axel Brunger'. **[software]**

_software.contact_author_email (*line*)
The e-mail address of the person specified in **_software.contact_author**.
Example: 'bourne@sdsc.edu'. **[software]**

_software.date (*line*)
The date the software was released.
Examples: '1991-10-01', '1990-04-30'. **[software]**

_software.dependencies (*line*)
Any prerequisite software required to run **_software.name**.
Example: 'PDBlib class library'. **[software]**

_software.description (*line*)
Description of the software.
Example: 'Uses method of restrained least squares'. **[software]**

_software.hardware (*line*)
The hardware upon which the software was run.
Examples: 'Sun Sparc 10 model 41', 'Dec Alpha 3000 model 500S', 'Silicon Graphics Elan', 'Compaq PC 486/66'. **[software]**

_software.language (*uline*)
The major computing language in which the software is coded.
The data value must be one of the following:
Ada
assembler
Awk
Basic
C++
C/C++
C
csh
Fortran
Fortran_77
'Fortran 77'
'Fortran 90'
Java
ksh
Pascal
Perl
Python
sh
Tcl
Other **[software]**

_software.location (*line*)
The URL for an Internet address at which details of the software can be found.
Examples:
'http://rosebud.sdsc.edu/projects/pb/IUCr/software.html',
'ftp://ftp.sdsc.edu/pub/sdsc/biology/'. **[software]**

_software.mods (*line*)
Any noteworthy modifications to the base software, if applicable.
Example: 'Added support for space group F432'. **[software]**

* **_software.name** (*text*)
The name of the software.
Examples: 'Merlot', 'O', 'Xengen', 'X-plor'. **[software]**

_software.os (*text*)
The name of the operating system under which the software runs.
Examples: 'Ultrix', 'OpenVMS', 'DOS', 'Windows 95', 'Windows NT', 'Irix', 'HPUX', 'DEC Unix'. **[software]**

_software.os_version (*text*)
The version of the operating system under which the software runs.
Examples: '3.1', '4.2.1'. **[software]**

_software.type (*uline*)

The classification of the software according to the most common types.

The data value must be one of the following:

program	individual program with limited functionality
library	used by a program at load time
package	collections of programs with multiple functionality
filter	filters input and output streams
jiffy	short, simple program
other	all other kinds of software

[software]

***_software.version** (*line*)

The version of the software.

Examples: 'v1.0', 'beta', '3.1-2', 'unknown'. **[software]**

SPACE_GROUP

Contains all the data items that refer to the space group as a whole, such as its name or crystal system. They may be looped, for example, in a list of space groups and their properties. Only a subset of the SPACE_GROUP category items appear in this dictionary. The remainder are found in the symmetry CIF dictionary. Space-group types are identified by their number as given in *International Tables for Crystallography* Vol. A. Specific settings of the space groups can be identified either by their Hall symbol or by specifying their symmetry operations. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but several different Hermann–Mauguin symbols may refer to the same space-group type. A Hermann–Mauguin symbol contains information on the choice of the basis, but not on the choice of origin. Different formats for the Hermann–Mauguin symbol are found in the symmetry CIF dictionary.

Category key(s): **_space_group.id**

Example 1 – the monoclinic space group No. 15 with unique axis b.

```
_space_group.id                 1
_space_group.name_H-M_alt       'C 2/c'
_space_group.IT_number          15
_space_group.name_Hall          '-C 2yc'
_space_group.crystal_system     monoclinic
```

_space_group.crystal_system (*code*)

_space_group_crystal_system (*cif_core.dic 2.3*)

The name of the system of geometric crystal classes of space groups (crystal system) to which the space group belongs. Note that rhombohedral space groups belong to the trigonal system.

Related item: **_symmetry.cell_setting** (alternate).

The data value must be one of the following:

triclinic
monoclinic
orthorhombic
tetragonal
trigonal
hexagonal
cubic

[space_group]

***_space_group.id** (*code*)

_space_group_id (*cif_core.dic 2.3*)

This is the unique identifier for the SPACE_GROUP category.

[space_group]

_space_group.IT_number (*int*)

_space_group_IT_number (*cif_core.dic 2.3*)

The number as assigned in *International Tables for Crystallography* Vol. A, specifying the proper affine class (*i.e.* the orientation-preserving affine class) of space groups (crystallographic space-group type) to which the space group belongs. This number defines the space-group type but not the coordinate system in which it is expressed.

The permitted range is [1, 230].

Related item: **_symmetry.Int_Tables_number** (alternate). **[space_group]**

_space_group.name_H-M_alt (*line*)

_space_group_name_H-M_alt (*cif_core.dic 2.3*)

_space_group.name_H-M_alt allows any Hermann–Mauguin symbol to be given. The way in which this item is used is determined by the user and in general is not intended to be interpreted by computer. It may, for example, be used to give one of the extended Hermann–Mauguin symbols given in Table 4.3.2.1 of *International Tables for Crystallography* Vol. A (2002) or a Hermann–Mauguin symbol for a conventional or unconventional setting. Each component of the space-group name is separated by a space or an underscore. The use of a space is strongly recommended. The underscore is only retained because it was used in old CIFs. It should not be used in new CIFs. Subscripts should appear without special symbols. Bars should be given as negative signs before the numbers to which they apply. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but a given space-group type may be described by more than one Hermann–Mauguin symbol. The space-group type is best described using **_space_group.IT_number**. The Hermann–Mauguin symbol may contain information on the choice of basis, but not on the choice of origin. To define the setting uniquely, use **_space_group.name_Hall** or list the symmetry operations.

Related item: **_symmetry.space_group_name_H-M** (alternate) .

Example:

```
; loop_
  _space_group.name_H-M_alt
  'C m c m'
  'C 2/c 2/m 21/m'
  'A m a m'
;  (three examples for space group No. 63)
```

[space_group]

_space_group.name_Hall (*line*)

_space_group_name_Hall (*cif_core.dic 2.3*)

Space-group symbol defined by Hall. Each component of the space-group name is separated by a space or an underscore. The use of a space is strongly recommended. The underscore is only retained because it was used in old CIFs. It should not be used in new CIFs. **_space_group.name_Hall** uniquely defines the space group and its reference to a particular coordinate system.

Reference: Hall, S. R. (1981). *Acta Cryst.* A**37**, 517–525; erratum (1981), A**37**, 921. [See also *International Tables for Crystallography* Vol. B (2001), Chapter 1.4, Appendix 1.4.2.]

Related item: **_symmetry.space_group_name_Hall** (alternate) .

Examples: '`P 2c -2ac`' (equivalent to $Pca2_1$), '`-I 4bd 2ab 3`' (equivalent to $Ia3d$).

[space_group]

SPACE_GROUP_SYMOP

Contains information about the symmetry operations of the space group.

Category key(s): **_space_group_symop.id**

Example 1 – The symmetry operations for the space group $P2_1/c$.

```
loop_
_space_group_symop.id
_space_group_symop.operation_xyz
  1    x,y,z
  2   -x,-y,-z
  3   -x,1/2+y,1/2-z
  4    x,1/2-y,1/2+z
```

*_space_group_symop.id (*code*)

_space_group_symop_id (*cif_core.dic 2.3*)

An arbitrary identifier that uniquely labels each symmetry operation in the list.

Related item: **_symmetry_equiv.id** (alternate). **[space_group_symop]**

_space_group_symop.operation_xyz (*line*)

_space_group_symop_operation_xyz (*cif_core.dic 2.3*)

A parsable string giving one of the symmetry operations of the space group in algebraic form. If W is a matrix representation of the rotational part of the symmetry operation defined by the positions and signs of x, y and z, and $\mathbf{w}$ is a column of translations defined by fractions, an equivalent position $\boldsymbol{x}'$ is generated from a given position $\boldsymbol{x}$ by

$$\boldsymbol{x}' = W\boldsymbol{x} + \mathbf{w}.$$

When a list of symmetry operations is given, it must contain a complete set of coordinate representatives which generates all the operations of the space group by the addition of all primitive translations of the space group. Such representatives are to be found as the coordinates of the general-equivalent position in *International Tables for Crystallography* Vol. A (2002), to which it is necessary to add any centring translations shown above the general-equivalent position. That is to say, it is necessary to list explicitly all the symmetry operations required to generate all the atoms in the unit cell defined by the setting used.

Related item: **_symmetry_equiv.pos_as_xyz** (alternate) .

Example: '`x,1/2-y,1/2+z`' (glide reflection through the plane (x, 1/4, z), with glide vector (1/2)**c**). **[space_group_symop]**

_space_group_symop.sg_id (*code*)

_space_group_symop_sg_id (*cif_core.dic 2.3*)

This must match a particular value of **_space_group.id**, allowing the symmetry operation to be identified with a particular space group.

[space_group_symop]

STRUCT

Data items in the STRUCT category record details about the description of the crystallographic structure.

Category group(s): **inclusive_group**
struct_group

Category key(s): **_struct.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_struct.entry_id                    '5HVP'
_struct.title
; HIV-1 protease complex with acetyl-pepstatin
;
```

*_struct.entry_id

This data item is a pointer to **_entry.id** in the ENTRY category.

_struct.title (*text*)

A title for the data block. The author should attempt to convey the essence of the structure archived in the CIF in the title, and to distinguish this structural result from others.

Examples: '`5'-D(*(I)CP*CP*GP*G)-3`', '`T4 lysozyme mutant - S32A`', '`hen egg white lysozyme at -30 degrees C`', '`quail egg white lysozyme at 2 atmospheres`'. **[struct]**

STRUCT_ASYM

Data items in the STRUCT_ASYM category record details about the structural elements in the asymmetric unit.

Category group(s): **inclusive_group**
struct_group

Category key(s): **_struct_asym.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_asym.id
_struct_asym.entity_id
_struct_asym.details
  A  1  'one monomer of the dimeric enzyme'
  B  1  'one monomer of the dimeric enzyme'
  C  2  'one partially occupied position for the inhibitor'
  D  2  'one partially occupied position for the inhibitor'
```

_struct_asym.details (*text*)

A description of special aspects of this portion of the contents of the asymmetric unit.

Example:

```
; The drug binds to this enzyme in two roughly twofold
symmetric modes. Hence this biological unit (3) is roughly
twofold symmetric to biological unit (2). Disorder in the
protein chain indicated with alternative ID 2 should be used
with this biological unit.
;
```

[struct_asym]

*_struct_asym.entity_id

This data item is a pointer to **_entity.id** in the ENTITY category.

*_struct_asym.id (*code*)

The value of **_struct_asym.id** must uniquely identify a record in the STRUCT_ASYM list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_atom_site.label_asym_id,
_struct_biol_gen.asym_id,
_geom_angle.atom_site_label_asym_id_1,
_geom_angle.atom_site_label_asym_id_2,
_geom_angle.atom_site_label_asym_id_3,
_geom_bond.atom_site_label_asym_id_1,
_geom_bond.atom_site_label_asym_id_2,
_geom_contact.atom_site_label_asym_id_1,
_geom_contact.atom_site_label_asym_id_2,
_geom_hbond.atom_site_label_asym_id_A,
_geom_hbond.atom_site_label_asym_id_D,
_geom_hbond.atom_site_label_asym_id_H,
_geom_torsion.atom_site_label_asym_id_1,
_geom_torsion.atom_site_label_asym_id_2,
_geom_torsion.atom_site_label_asym_id_3,
_geom_torsion.atom_site_label_asym_id_4,
_struct_conf.beg_label_asym_id,
_struct_conf.end_label_asym_id,
_struct_conn.ptnr1_label_asym_id,

_struct_conn.ptnr2_label_asym_id,
_struct_mon_nucl.label_asym_id,
_struct_mon_prot.label_asym_id,
_struct_mon_prot_cis.label_asym_id,
_struct_ncs_dom_lim.beg_label_asym_id,
_struct_ncs_dom_lim.end_label_asym_id,
_struct_sheet_range.beg_label_asym_id,
_struct_sheet_range.end_label_asym_id,
_struct_site_gen.label_asym_id.

Examples: '1', 'A', '2B3'. **[struct_asym]**

STRUCT_BIOL

Data items in the STRUCT_BIOL category record details about the structural elements that form each structure of biological significance. A given crystal structure may contain many different biological structures. A given structural component in the asymmetric unit may be part of more than one biological unit. A given biological structure may involve crystallographic symmetry. For instance, in a structure of a lysozyme–FAB structure, the light- and heavy-chain components of the FAB could be one biological unit, while the two chains of the FAB and the lysozyme could constitute a second biological unit.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_biol.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_biol.id
_struct_biol.details
  1
; significant deviations from twofold symmetry exist in this
  dimeric enzyme
;
  2
; The drug binds to this enzyme in two roughly twofold
  symmetric modes. Hence this biological unit (2) is roughly
  twofold symmetric to biological unit (3). Disorder in the
  protein chain indicated with alternative ID 1 should be
  used with this biological unit.
;
  3
; The drug binds to this enzyme in two roughly twofold
  symmetric modes. Hence this biological unit (3) is roughly
  twofold symmetric to biological unit (2). Disorder in the
  protein chain indicated with alternative ID 2 should be
  used with this biological unit.
;
```

_struct_biol.details (*text*)

A description of special aspects of the biological unit.

Example:

```
; The drug binds to this enzyme in two roughly twofold
symmetric modes. Hence this biological unit (3) is roughly
twofold symmetric to biological unit (2). Disorder in the
protein chain indicated with alternative ID 2 should be used
with this biological unit.
;
```
[struct_biol]

*_struct_biol.id (*line*)

The value of **_struct_biol.id** must uniquely identify a record in the STRUCT_BIOL list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_struct_biol_gen.biol_id,
_struct_biol_keywords.biol_id,
_struct_biol_view.biol_id,
_struct_ref.biol_id. **[struct_biol]**

STRUCT_BIOL_GEN

Data items in the STRUCT_BIOL_GEN category record details about the generation of each biological unit. The STRUCT_BIOL_GEN data items provide the specifications of the components that constitute that biological unit, which may include symmetry elements.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_biol_gen.biol_id**
_struct_biol_gen.asym_id
_struct_biol_gen.symmetry

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_biol_gen.biol_id
_struct_biol_gen.asym_id
_struct_biol_gen.symmetry
  1  A  1_555
  1  B  1_555
  2  A  1_555
  2  B  1_555
  2  C  1_555
  3  A  1_555
  3  B  1_555
  3  D  1_555
```

*_struct_biol_gen.asym_id

This data item is a pointer to **_struct_asym.id** in the STRUCT_ASYM category.

*_struct_biol_gen.biol_id

This data item is a pointer to **_struct_biol.id** in the STRUCT_BIOL category.

_struct_biol_gen.details (*text*)

A description of special aspects of the symmetry generation of this portion of the biological structure.

Example:

```
; The zinc atom lies on a special position; application of
symmetry elements to generate the insulin hexamer will
generate excess zinc atoms, which must be removed by hand.
;
```
[struct_biol_gen]

*_struct_biol_gen.symmetry (*symop*)

Describes the symmetry operation that should be applied to the atom set specified by **_struct_biol_gen.asym_id** to generate a portion of the biological structure.

Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[struct_biol_gen]**

STRUCT_BIOL_KEYWORDS

Data items in the STRUCT_BIOL_KEYWORDS category record keywords that describe each biological unit.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_biol_keywords.biol_id**
_struct_biol_keywords.text

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_biol_keywords.biol_id
_struct_biol_keywords.text
  1  'aspartyl-protease'
  1  'aspartic-protease'
  1  'acid-protease'
  1  'aspartyl-proteinase'
  1  'aspartic-proteinase'
  1  'acid-proteinase'
  1  'enzyme'
  1  'protease'
  1  'proteinase'
  1  'dimer'
  2  'drug-enzyme complex'
  2  'inhibitor-enzyme complex'
  2  'drug-protease complex'
  2  'inhibitor-protease complex'
  3  'drug-enzyme complex'
  3  'inhibitor-enzyme complex'
  3  'drug-protease complex'
  3  'inhibitor-protease complex'
```

* **_struct_biol_keywords.biol_id**

This data item is a pointer to **_struct_biol.id** in the STRUCT_BIOL category.

* **_struct_biol_keywords.text** (*text*)

Keywords describing this biological entity.

Examples: 'antibody', 'antigen', 'enzyme', 'cytokine', 'tRNA'.

[struct_biol_keywords]

STRUCT_BIOL_VIEW

Data items in the STRUCT_BIOL_VIEW category record details about how to draw and annotate an informative view of the biological structure.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_biol_view.biol_id**
_struct_biol_view.id

*Example 1 – based on NDB structure GDL001 by Coll, Aymami, Van Der Marel, Van Boom, Rich & Wang [Biochemistry, (1989), **28**, 310–320].*

```
_struct_biol_view.biol_id              c1
_struct_biol_view.id                   1
_struct_biol_view.rot_matrix[1][1]     0.132
_struct_biol_view.rot_matrix[1][2]     0.922
_struct_biol_view.rot_matrix[1][3]    -0.363
_struct_biol_view.rot_matrix[2][1]     0.131
_struct_biol_view.rot_matrix[2][2]    -0.380
_struct_biol_view.rot_matrix[2][3]    -0.916
_struct_biol_view.rot_matrix[3][1]    -0.982
_struct_biol_view.rot_matrix[3][2]     0.073
_struct_biol_view.rot_matrix[3][3]    -0.172
_struct_biol_view.details
; This view highlights the ATAT-Netropsin interaction in the
  DNA-drug complex.
;
```

* **_struct_biol_view.biol_id**

This data item is a pointer to **_struct_biol.id** in the STRUCT_BIOL category.

_struct_biol_view.details (*text*)

A description of special aspects of this view of the biological structure. This data item can be used as a figure legend.

Example:

```
; The enzyme has been oriented with the molecular twofold axis
aligned with the horizontal axis of the figure.
;
```

[struct_biol_view]

* **_struct_biol_view.id** (*line*)

The value of **_struct_biol_view.id** must uniquely identify a record in the STRUCT_BIOL_VIEW list. Note that this item need not be a number; it can be any unique identifier.

Examples: 'Figure 1', 'unliganded enzyme', 'view down enzyme active site'.

[struct_biol_view]

_struct_biol_view.rot_matrix[1][1] (*float*)

The [1][1] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[1][2] (*float*)

The [1][2] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[1][3] (*float*)

The [1][3] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}}$$

[struct_biol_view]

_struct_biol_view.rot_matrix[2][1] (*float*)

The [2][1] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[2][2] (*float*)
The [2][2] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[2][3] (*float*)
The [2][3] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[3][1] (*float*)
The [3][1] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[3][2] (*float*)
The [3][2] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

_struct_biol_view.rot_matrix[3][3] (*float*)
The [3][3] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_BIOL_GEN category to give a view useful for describing the structure. The conventions used in the rotation are described in **_struct_biol_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_biol_view]

STRUCT_CONF

Data items in the STRUCT_CONF category record details about the backbone conformation of a segment of polymer. Data items in the STRUCT_CONF_TYPE category define the criteria used to identify the backbone conformations.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_conf.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_conf.id
_struct_conf.conf_type_id
_struct_conf.beg_label_comp_id
_struct_conf.beg_label_asym_id
_struct_conf.beg_label_seq_id
_struct_conf.end_label_comp_id
_struct_conf.end_label_asym_id
_struct_conf.end_label_seq_id
_struct_conf.details
  HELX1  HELX_RH_AL_P  ARG  A   87  GLN  A   92  .
  HELX2  HELX_RH_AL_P  ARG  B  287  GLN  B  292  .
  STRN1  STRN          PRO  A    1  LEU  A    5  .
  STRN2  STRN          CYS  B  295  PHE  B  299  .
  STRN3  STRN          CYS  A   95  PHE  A  299  .
  STRN4  STRN          PRO  B  201  LEU  B  205  .
# - - - - data truncated for brevity - - - -
  TURN1  TURN_TY1P_P   ILE  A   15  GLN  A   18  .
  TURN2  TURN_TY2_P    GLY  A   49  GLY  A   52  .
  TURN3  TURN_TY1P_P   ILE  A   55  HIS  A   69  .
  TURN4  TURN_TY1_P    THR  A   91  GLY  A   94  .
# - - - - data truncated for brevity - - - -
```

_struct_conf.beg_auth_asym_id
A component of the identifier for the residue at which the conformation segment begins. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_conf.beg_auth_comp_id
A component of the identifier for the residue at which the conformation segment begins. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_conf.beg_auth_seq_id
A component of the identifier for the residue at which the conformation segment begins. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

***_struct_conf.beg_label_asym_id**
A component of the identifier for the residue at which the conformation segment begins. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_conf.beg_label_comp_id**
A component of the identifier for the residue at which the conformation segment begins. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_conf.beg_label_seq_id**
A component of the identifier for the residue at which the conformation segment begins. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

***_struct_conf.conf_type_id**
This data item is a pointer to **_struct_conf_type.id** in the STRUCT_CONF_TYPE category.

_struct_conf.details (*text*)
A description of special aspects of the conformation assignment.

[struct_conf]

_struct_conf.end_auth_asym_id
A component of the identifier for the residue at which the conformation segment ends. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_conf.end_auth_comp_id
A component of the identifier for the residue at which the conformation segment ends. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_conf.end_auth_seq_id
A component of the identifier for the residue at which the conformation segment ends. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

***_struct_conf.end_label_asym_id**
A component of the identifier for the residue at which the conformation segment ends. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_conf.end_label_comp_id**
A component of the identifier for the residue at which the conformation segment ends. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_conf.end_label_seq_id**
A component of the identifier for the residue at which the conformation segment ends. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

***_struct_conf.id** (*code*)
The value of **_struct_conf.id** must uniquely identify a record in the STRUCT_CONF list. Note that this item need not be a number; it can be any unique identifier.

[struct_conf]

STRUCT_CONF_TYPE

Data items in the STRUCT_CONF_TYPE category record details about the criteria used to identify backbone conformations of a segment of polymer.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_conf_type.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_conf_type.id
_struct_conf_type.criteria
_struct_conf_type.reference
  HELX_RH_AL_P  'author judgement'  .
  STRN          'author judgement'  .
  TURN_TY1_P    'author judgement'  .
  TURN_TY1P_P   'author judgement'  .
  TURN_TY2_P    'author judgement'  .
  TURN_TY2P_P   'author judgement'  .
```

***_struct_conf_type.id** (*ucode*)
The descriptor that categorizes the type of the conformation of the backbone of the polymer (whether protein or nucleic acid). Explicit values for the torsion angles that define each conformation are not given here, but it is expected that the author would provide such information in either the **_struct_conf_type.criteria** or **_struct_conf_type.reference** data items, or both.

The following item(s) have an equivalent role in their respective categories:
_struct_conf.conf_type_id.

The data value must be one of the following:

HELX_P	helix with handedness and type not specified (protein)
HELX_OT_P	helix with handedness and type that do not conform to an accepted category (protein)
HELX_RH_P	right-handed helix with type not specified (protein)
HELX_RH_OT_P	right-handed helix with type that does not conform to an accepted category (protein)
HELX_RH_AL_P	right-handed α helix (protein)
HELX_RH_GA_P	right-handed γ helix (protein)
HELX_RH_OM_P	right-handed ω helix (protein)
HELX_RH_PI_P	right-handed π helix (protein)
HELX_RH_27_P	right-handed 2–7 helix (protein)
HELX_RH_3T_P	right-handed 3–10 helix (protein)
HELX_RH_PP_P	right-handed polyproline helix (protein)
HELX_LH_P	left-handed helix with type not specified (protein)
HELX_LH_OT_P	left-handed helix with type that does not conform to an accepted category (protein)
HELX_LH_AL_P	left-handed α helix (protein)
HELX_LH_GA_P	left-handed γ helix (protein)
HELX_LH_OM_P	left-handed ω helix (protein)
HELX_LH_PI_P	left-handed π helix (protein)
HELX_LH_27_P	left-handed 2–7 helix (protein)
HELX_LH_3T_P	left-handed 3–10 helix (protein)
HELX_LH_PP_P	left-handed polyproline helix (protein)
HELX_N	helix with handedness and type not specified (nucleic acid)
HELX_OT_N	helix with handedness and type that do not conform to an accepted category (nucleic acid)
HELX_RH_N	right-handed helix with type not specified (nucleic acid)
HELX_RH_OT_N	right-handed helix with type that does not conform to an accepted category (nucleic acid)
HELX_RH_A_N	right-handed A helix (nucleic acid)
HELX_RH_B_N	right-handed B helix (nucleic acid)
HELX_RH_Z_N	right-handed Z helix (nucleic acid)
HELX_LH_N	left-handed helix with type not specified (nucleic acid)
HELX_LH_OT_N	left-handed helix with type that does not conform to an accepted category (nucleic acid)
HELX_LH_A_N	left-handed A helix (nucleic acid)
HELX_LH_B_N	left-handed B helix (nucleic acid)
HELX_LH_Z_N	left-handed Z helix (nucleic acid)
TURN_P	turn with type not specified (protein)
TURN_OT_P	turn with type that does not conform to an accepted category (protein)
TURN_TY1_P	type I turn (protein)
TURN_TY1P_P	type I′ turn (protein)
TURN_TY2_P	type II turn (protein)
TURN_TY2P_P	type II′ turn (protein)
TURN_TY3_P	type III turn (protein)
TURN_TY3P_P	type III′ turn (protein)
STRN	β strand (protein)

[struct_conf_type]

_struct_conf_type.criteria (*text*)
The criteria used to assign this conformation type.

Examples: 'author judgement', 'phi=54-74, psi=30-50'.

[struct_conf_type]

_struct_conf_type.reference (*text*)
A literature reference that defines the criteria used to assign this conformation type and subtype.

[struct_conf_type]

STRUCT_CONN

Data items in the STRUCT_CONN category record details about the connections between portions of the structure. These can be hydrogen bonds, salt bridges, disulfide bridges and so on. The STRUCT_CONN_TYPE records define the criteria used to identify these connections.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_conn.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_conn.id
_struct_conn.conn_type_id
_struct_conn.ptnr1_label_comp_id
_struct_conn.ptnr1_label_asym_id
_struct_conn.ptnr1_label_seq_id
_struct_conn.ptnr1_label_atom_id
_struct_conn.ptnr1_role
_struct_conn.ptnr1_symmetry
_struct_conn.ptnr2_label_comp_id
_struct_conn.ptnr2_label_asym_id
_struct_conn.ptnr2_label_seq_id
_struct_conn.ptnr2_label_atom_id
_struct_conn.ptnr2_role
_struct_conn.ptnr2_symmetry
_struct_conn.details
C1  saltbr  ARG  A  87 NZ1 positive 1_555 GLU  A  92  OE1
     negative 1_555  .
C2  hydrog  ARG  B 287 N   donor    1_555 GLY  B 292  O
     acceptor 1_555  .
# - - - - data truncated for brevity - - - -
```

***_struct_conn.conn_type_id**
This data item is a pointer to **_struct_conn_type.id** in the STRUCT_CONN_TYPE category.

_struct_conn.details (*text*)
A description of special aspects of the connection.
Example: 'disulfide bridge C-S-S-C is highly distorted'.
[struct_conn]

***_struct_conn.id** (*code*)
The value of **_struct_conn.id** must uniquely identify a record in the STRUCT_CONN list. Note that this item need not be a number; it can be any unique identifier.
[struct_conn]

_struct_conn.ptnr1_auth_asym_id
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_conn.ptnr1_auth_atom_id
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_conn.ptnr1_auth_comp_id
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_conn.ptnr1_auth_seq_id
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_struct_conn.ptnr1_label_alt_id
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_conn.ptnr1_label_asym_id**
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_conn.ptnr1_label_atom_id**
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_struct_conn.ptnr1_label_comp_id**
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_conn.ptnr1_label_seq_id**
A component of the identifier for partner 1 of the structure connection. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_conn.ptnr1_role (*uline*)
The chemical or structural role of the first partner in the structure connection.
Examples: 'donor', 'acceptor', 'negative', 'positive', 'metal', 'metal coordination'. **[struct_conn]**

_struct_conn.ptnr1_symmetry (*symop*)
Describes the symmetry operation that should be applied to the atom set specified by **_struct_conn.ptnr1_label*** to generate the first partner in the structure connection.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[struct_conn]**

_struct_conn.ptnr2_auth_asym_id
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_conn.ptnr2_auth_atom_id
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_conn.ptnr2_auth_comp_id
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_conn.ptnr2_auth_seq_id
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_struct_conn.ptnr2_label_alt_id
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_conn.ptnr2_label_asym_id**
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_conn.ptnr2_label_atom_id**
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_struct_conn.ptnr2_label_comp_id**
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_conn.ptnr2_label_seq_id**
A component of the identifier for partner 2 of the structure connection. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_conn.ptnr2_role (*uline*)
The chemical or structural role of the second partner in the structure connection.
Examples: 'donor', 'acceptor', 'negative', 'positive', 'metal', 'metal coordination'. **[struct_conn]**

_struct_conn.ptnr2_symmetry (*symop*)
Describes the symmetry operation that should be applied to the atom set specified by **_struct_conn.ptnr2_label*** to generate the second partner in the structure connection.
Examples: '.' (no symmetry or translation to site), '4' (4th symmetry operation applied), '7_645' (7th symmetry position: $+a$ on x, $-b$ on y). **[struct_conn]**

STRUCT_CONN_TYPE

Data items in the STRUCT_CONN_TYPE category record details about the criteria used to identify interactions between portions of the structure.
Category group(s): **inclusive_group**, **struct_group**
Category key(s): **_struct_conn_type.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_conn_type.id
_struct_conn_type.criteria
_struct_conn_type.reference
  saltbr
 'negative to positive distance > 2.5 \%A, < 3.2 \%A'      .
  hydrog
 'NO distance > 2.5\%A, < 3.5\%A, NOC angle < 120 degrees' .
```

_struct_conn_type.criteria (*text*)
The criteria used to define the interaction.
Examples: 'O to N distance > 2.5 \%A, < 3.2 \%A', 'authors judgement'. **[struct_conn_type]**

***_struct_conn_type.id** (*ucode*)
The chemical or structural type of the interaction.
The following item(s) have an equivalent role in their respective categories:
_struct_conn.conn_type_id.
The data value must be one of the following:

covale	covalent bond
disulf	disulfide bridge
hydrog	hydrogen bond
metalc	metal coordination
mismat	mismatched base pairs
saltbr	ionic interaction
modres	covalent residue modification
covale_base	covalent modification of a nucleotide base
covale_sugar	covalent modification of a nucleotide sugar
covale_phosphate	covalent modification of a nucleotide phosphate

[struct_conn_type]

_struct_conn_type.reference (*text*)
A reference that specifies the criteria used to define the interaction. **[struct_conn_type]**

STRUCT_KEYWORDS

Data items in the STRUCT_KEYWORDS category specify keywords that describe the chemical structure in this entry.
Category group(s): **inclusive_group**, **struct_group**
Category key(s): **_struct_keywords.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_keywords.entry_id
_struct_keywords.text
 '5HVP' 'enzyme-inhibitor complex'
 '5HVP' 'aspartyl protease'
 '5HVP' 'structure-based drug design'
 '5HVP' 'static disorder'
```

***_struct_keywords.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_struct_keywords.text (*text*)
Keywords describing this structure.
Examples: 'serine protease', 'inhibited complex', 'high-resolution refinement'. **[struct_keywords]**

STRUCT_MON_DETAILS

Data items in the STRUCT_MON_DETAILS category record details about specifics of calculations summarized in data items in the STRUCT_MON_PROT and STRUCT_MON_NUCL categories. These can include the coefficients used in map calculations, the radii used for including points in a calculation and so on.
Category group(s): **inclusive_group**, **struct_group**
Category key(s): **_struct_mon_details.entry_id**

***_struct_mon_details.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_struct_mon_details.prot_cis (*float*)
An ideal *cis* peptide bond would have an ω torsion angle of zero. This data item gives the value in degrees by which the observed torsion angle can differ from 0.0 and still be considered *cis*.
Example: '30.0'. **[struct_mon_details]**

_struct_mon_details.RSCC (*text*)

This data item describes the specifics of the calculations that generated the values given in **_struct_mon_prot.RSCC_all**, **_struct_mon_prot.RSCC_main** and **_struct_mon_prot.RSCC_side**. The coefficients used to calculate the $p(o)$ and $p(c)$ maps should be given as well as the criterion for the inclusion of map grid points in the calculation.

Examples:

```
; The map p(o) was calculated with coefficients
  2F(o) - F(c) and with phase alpha(c). F(o)
  are the observed structure-factor amplitudes,
  F(c) are the amplitudes calculated from the
  current model and alpha(c) are the phases
  calculated from the current model.
  The map p(c) was calculated in program O using
  a Gaussian distribution function around the
  atoms in the current model.
  Map grid points within 1.5 A of the
  designated atoms were included in the
  calculation.
;
; The map p(o) was calculated with coefficients
  F(o) and with phase alpha(c). F(o) are the
  observed structure-factor amplitudes, and
  alpha(c) are the phases calculated from the
  current model.
  The map p(c) was calculated with coefficients
  F(c) and with phases alpha(c). F(c) and
  alpha(c) are the structure-factor amplitudes
  and phases, respectively, calculated from the
  current model.
  Map grid points within a van der Waals radius
  of the designated atoms were included in the
  calculation.
;
```

[struct_mon_details]

_struct_mon_details.RSR (*text*)

This data item describes the specifics of the calculations that generated the values given in **_struct_mon_prot.RSR_all**, **_struct_mon_prot.RSR_main** and **_struct_mon_prot.RSR_side**. The coefficients used to calculate the $p(o)$ and $p(c)$ maps should be given as well as the criterion for the inclusion of map grid points in the calculation.

Examples:

```
; The map p(o) was calculated with coefficients
  2F(o) - F(c) and with phase alpha(c). F(o)
  are the observed structure-factor amplitudes,
  F(c) are the amplitudes calculated from the
  current model and alpha(c) are the phases
  calculated from the current model.
  The map p(c) was calculated in program O using
  a Gaussian distribution function around the
  atoms in the current model.
  Map grid points within 1.5 A of the
  designated atoms were included in the
  calculation.
;
; The map p(o) was calculated with coefficients
  F(o) and with phase alpha(c). F(o) are the
  observed structure-factor amplitudes, and
  alpha(c) are the phases calculated from the
  current model.
  The map p(c) was calculated with coefficients
  F(c) and with phases alpha(c). F(c) and
  alpha(c) are the structure-factor amplitudes
  and phases, respectively, calculated from the
  current model.
  Map grid points within a van der Waals radius
  of the designated atoms were included in the
  calculation.
;
```

[struct_mon_details]

STRUCT_MON_NUCL

Data items in the STRUCT_MON_NUCL category record details about structural properties of a nucleic acid when analyzed at the monomer level. Analogous data items for proteins are given in the STRUCT_MON_PROT category. For items where the value of the property depends on the method employed to calculate it, details of the method of calculation are given using data items in the STRUCT_MON_DETAILS category.

Category group(s): **inclusive_group**
struct_group

Category key(s): **_struct_mon_nucl.label_alt_id**
_struct_mon_nucl.label_asym_id
_struct_mon_nucl.label_comp_id
_struct_mon_nucl.label_seq_id

Example 1 – based on NDB structure BDL028.

```
loop_
_struct_mon_nucl.label_comp_id
_struct_mon_nucl.label_seq_id
_struct_mon_nucl.label_asym_id
_struct_mon_nucl.label_alt_id
_struct_mon_nucl.alpha
_struct_mon_nucl.beta
_struct_mon_nucl.gamma
_struct_mon_nucl.delta
_struct_mon_nucl.epsilon
_struct_mon_nucl.zeta
C  1 A .      .      .   29.9  131.9  222.1  174.2
G  2 A .  334.0  130.6   33.1  125.6  167.6  270.9
T  3 A .  258.2  178.7   101.0 114.6  216.6  259.3
#         ---- abbreviated list -----
```

_struct_mon_nucl.alpha (*float*)

The value in degrees of the backbone torsion angle α (O3′—P—O5′—C5′).

[struct_mon_nucl]

_struct_mon_nucl.auth_asym_id

A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_mon_nucl.auth_comp_id

A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_mon_nucl.auth_seq_id

A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_struct_mon_nucl.beta (*float*)

The value in degrees of the backbone torsion angle β (P—O5′—C5′—C4′).

[struct_mon_nucl]

_struct_mon_nucl.chi1 (*float*)

The value in degrees of the sugar–base torsion angle χ_1 (O4′—C1′—N1—C2).

[struct_mon_nucl]

_struct_mon_nucl.chi2 (*float*)

The value in degrees of the sugar–base torsion angle χ_2 (O4′—C1′—N9—C4).

[struct_mon_nucl]

_struct_mon_nucl.delta (*float*)
The value in degrees of the backbone torsion angle δ (C5′—C4′—C3′—O3′).

[struct_mon_nucl]

_struct_mon_nucl.details (*float*)
A description of special aspects of the residue, its conformation, behaviour in refinement, or any other aspect that requires annotation.

Example:
```
; Part of the phosphodiester backbone not in density.
;
```
[struct_mon_nucl]

_struct_mon_nucl.epsilon (*float*)
The value in degrees of the backbone torsion angle ε (C4′—C3′—O3′—P).

[struct_mon_nucl]

_struct_mon_nucl.gamma (*float*)
The value in degrees of the backbone torsion angle γ (O5′—C5′—C4′—C3′).

[struct_mon_nucl]

***_struct_mon_nucl.label_alt_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_mon_nucl.label_asym_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_mon_nucl.label_comp_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_mon_nucl.label_seq_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_mon_nucl.mean_B_all (*float*)
The mean value of the isotropic displacement parameter for all atoms in the monomer.

[struct_mon_nucl]

_struct_mon_nucl.mean_B_base (*float*)
The mean value of the isotropic displacement parameter for atoms in the base moiety of the nucleic acid monomer.

[struct_mon_nucl]

_struct_mon_nucl.mean_B_phos (*float*)
The mean value of the isotropic displacement parameter for atoms in the phosphate moiety of the nucleic acid monomer.

[struct_mon_nucl]

_struct_mon_nucl.mean_B_sugar (*float*)
The mean value of the isotropic displacement parameter for atoms in the sugar moiety of the nucleic acid monomer.

[struct_mon_nucl]

_struct_mon_nucl.nu0 (*float*)
The value in degrees of the sugar torsion angle ν_0 (C4′—O4′—C1′—C2′).

[struct_mon_nucl]

_struct_mon_nucl.nu1 (*float*)
The value in degrees of the sugar torsion angle ν_1 (O4′—C1′—C2′—C3′).

[struct_mon_nucl]

_struct_mon_nucl.nu2 (*float*)
The value in degrees of the sugar torsion angle ν_2 (C1′—C2′—C3′—C4′).

[struct_mon_nucl]

_struct_mon_nucl.nu3 (*float*)
The value in degrees of the sugar torsion angle ν_3 (C2′—C3′—C4′—O4′).

[struct_mon_nucl]

_struct_mon_nucl.nu4 (*float*)
The value in degrees of the sugar torsion angle ν_4 (C3′—C4′—O4′—C1′).

[struct_mon_nucl]

_struct_mon_nucl.P (*float*)
P is the phase angle of pseudorotation for five-membered rings. For ribose and deoxyribose sugars in nucleic acids

$$P = \arctan\left(\frac{(\tau_4+\tau_1)-(\tau_3+\tau_0)}{2\tau_2(\sin 36+\sin 72)}\right).$$

If τ_2 is < 0, then $P = P + 180^\circ$ (Altona & Sundaralingam, 1972).

Reference: Altona, C. & Sundaralingam, M. (1972). *J. Am. Chem. Soc.* **94**, 8205–8212.

[struct_mon_nucl]

_struct_mon_nucl.RSCC_all (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the nucleic acid monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|}{(\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|^2)^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.

[struct_mon_nucl]

_struct_mon_nucl.RSCC_base (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the base moiety of the nucleic acid monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}} \rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}} \rangle|}{(\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}} \rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}} \rangle|^2)^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.

[struct_mon_nucl]

_struct_mon_nucl.RSCC_phos (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the phosphate moiety of the nucleic acid monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}} \rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}} \rangle|}{(\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}} \rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}} \rangle|^2)^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.

[struct_mon_nucl]

_struct_mon_nucl.RSCC_sugar (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the sugar moiety of the nucleic acid monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}} \rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}} \rangle|}{(\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}} \rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}} \rangle|^2)^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.

[struct_mon_nucl]

_struct_mon_nucl.RSR_all (*float*)
The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the nucleic acid monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_nucl]

_struct_mon_nucl.RSR_base (*float*)
The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the base moiety of the nucleic acid monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_nucl]

_struct_mon_nucl.RSR_phos (*float*)
The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the phosphate moiety of the nucleic acid monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_nucl]

_struct_mon_nucl.RSR_sugar (*float*)
The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the sugar moiety of the nucleic acid monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_nucl]

_struct_mon_nucl.tau0 (*float*)
The value in degrees of the sugar torsion angle τ_0 (C4′—O4′—C1′—C2′).

[struct_mon_nucl]

_struct_mon_nucl.tau1 (*float*)
The value in degrees of the sugar torsion angle τ_1 (O4′—C1′—C2′—C3′).

[struct_mon_nucl]

_struct_mon_nucl.tau2 (*float*)
The value in degrees of the sugar torsion angle τ_2 (C1′—C2′—C3′—C4′).

[struct_mon_nucl]

_struct_mon_nucl.tau3 (*float*)
The value in degrees of the sugar torsion angle τ_3 (C2′—C3′—C4′—O4′).

[struct_mon_nucl]

_struct_mon_nucl.tau4 (*float*)
The value in degrees of the sugar torsion angle τ_4 (C3′—C4′—O4′—C1′).

[struct_mon_nucl]

_struct_mon_nucl.taum (*float*)
The maximum amplitude of puckering. This is derived from the pseudorotation value P and the torsion angles in the ribose ring.

$$\tau_2 = \tau_m \cos P,$$
$$\tau_3 = \tau_m \cos(P + 144),$$
$$\tau_4 = \tau_m \cos(P + 288),$$
$$\tau_0 = \tau_m \cos(P + 72),$$
$$\tau_1 = \tau_m \cos(P + 216).$$

[struct_mon_nucl]

_struct_mon_nucl.zeta (*float*)
The value in degrees of the backbone torsion angle ζ (C3′—O3′—P—O5′).

[struct_mon_nucl]

STRUCT_MON_PROT

Data items in the STRUCT_MON_PROT category record details about structural properties of a protein when analyzed at the monomer level. Analogous data items for nucleic acids are given in the STRUCT_MON_NUCL category. For items where the value of the property depends on the method employed to calculate it, details of the method of calculation are given using data items in the STRUCT_MON_DETAILS category.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_mon_prot.label_alt_id**
_struct_mon_prot.label_asym_id
_struct_mon_prot.label_comp_id
_struct_mon_prot.label_seq_id

Example 1 – based on laboratory records for protein NS1. This example provides details for residue ARG 35.

```
_struct_mon_prot.label_comp_id   ARG
_struct_mon_prot.label_seq_id     35
_struct_mon_prot.label_asym_id     A
_struct_mon_prot.label_alt_id      .
_struct_mon_prot.chi1           -67.9
_struct_mon_prot.chi2          -174.7
_struct_mon_prot.chi3           -67.7
_struct_mon_prot.chi4           -86.3
_struct_mon_prot.chi5             4.2
_struct_mon_prot.RSCC_all         0.90
_struct_mon_prot.RSR_all          0.18
_struct_mon_prot.mean_B_all       30.0
_struct_mon_prot.mean_B_main      25.0
_struct_mon_prot.mean_B_side      35.1
_struct_mon_prot.omega           180.1
_struct_mon_prot.phi             -60.3
_struct_mon_prot.psi             -46.0
```

_struct_mon_prot.auth_asym_id
A component of the identifier for the monomer. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_mon_prot.auth_comp_id
A component of the identifier for the monomer. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_mon_prot.auth_seq_id
A component of the identifier for the monomer. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_struct_mon_prot.chi1 (*float*)
The value in degrees of the side-chain torsion angle χ_1, for those residues containing such an angle.

[struct_mon_prot]

_struct_mon_prot.chi2 (*float*)
The value in degrees of the side-chain torsion angle χ_2, for those residues containing such an angle.

[struct_mon_prot]

_struct_mon_prot.chi3 (*float*)
The value in degrees of the side-chain torsion angle χ_3, for those residues containing such an angle.

[struct_mon_prot]

_struct_mon_prot.chi4 (*float*)
The value in degrees of the side-chain torsion angle χ_4, for those residues containing such an angle.

[struct_mon_prot]

_struct_mon_prot.chi5 (*float*)
The value in degrees of the side-chain torsion angle χ_5, for those residues containing such an angle.

[struct_mon_prot]

_struct_mon_prot.details (*float*)
A description of special aspects of the residue, its conformation, behaviour in refinement, or any other aspect that requires annotation.

Examples: 'very poor density',
```
; The side chain of this density may occupy alternative
conformations, but alternative conformations were not fit
in this model.
;
; This residue has a close contact with the bound inhibitor,
which may account for the nonstandard conformation of the
side chain.
;
```
[struct_mon_prot]

***_struct_mon_prot.label_alt_id**
A component of the identifier for the monomer. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_mon_prot.label_asym_id**
A component of the identifier for the monomer. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_mon_prot.label_comp_id**
A component of the identifier for the monomer. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_mon_prot.label_seq_id**
A component of the identifier for the monomer. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_mon_prot.mean_B_all (*float*)
The mean value of the isotropic displacement parameter for all atoms in the monomer.
[struct_mon_prot]

_struct_mon_prot.mean_B_main (*float*)
The mean value of the isotropic displacement parameter for atoms in the main chain of the monomer.
[struct_mon_prot]

_struct_mon_prot.mean_B_side (*float*)
The mean value of the isotropic displacement parameter for atoms in the side chain of the monomer.
[struct_mon_prot]

_struct_mon_prot.omega (*float*)
The value in degrees of the main-chain torsion angle ω.
[struct_mon_prot]

_struct_mon_prot.phi (*float*)
The value in degrees of the main-chain torsion angle φ.
[struct_mon_prot]

_struct_mon_prot.psi (*float*)
The value in degrees of the main-chain torsion angle ψ.
[struct_mon_prot]

_struct_mon_prot.RSCC_all (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|}{(\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|^2)^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.
[struct_mon_prot]

_struct_mon_prot.RSCC_main (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the main chain of the monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|}{[\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|^2]^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.
[struct_mon_prot]

_struct_mon_prot.RSCC_side (*float*)
The real-space (linear) correlation coefficient RSCC, as described by Jones *et al.* (1991), evaluated over all atoms in the side chain of the monomer.

$$\mathrm{RSCC} = \frac{\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle| \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|}{[\sum |p_{\mathrm{obs}} - \langle p_{\mathrm{obs}}\rangle|^2 \sum |p_{\mathrm{calc}} - \langle p_{\mathrm{calc}}\rangle|^2]^{1/2}},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSCC**. $\langle\rangle$ indicates an average and the sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSCC**.

Reference: Jones, T. A., Zou, J.-Y., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.
[struct_mon_prot]

_struct_mon_prot.RSR_all (*float*)
The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken

over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_prot]

_struct_mon_prot.RSR_main (*float*)

The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the main chain of the monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_prot]

_struct_mon_prot.RSR_side (*float*)

The real-space residual RSR, as described by Brändén & Jones (1990), evaluated over all atoms in the side chain of the monomer.

$$\mathrm{RSR} = \frac{\sum |p_{\mathrm{obs}} - p_{\mathrm{calc}}|}{\sum |p_{\mathrm{obs}} + p_{\mathrm{calc}}|},$$

where p_{obs} = the density in an 'experimental' map, p_{calc} = the density in a 'calculated' map and the sum is taken over the specified grid points. Details of how these maps were calculated should be given in **_struct_mon_details.RSR**. The sums are taken over all map grid points near the relevant atoms. The radius for including grid points in the calculation should also be given in **_struct_mon_details.RSR**.

Reference: Brändén, C.-I. & Jones, T. A. (1990). *Nature (London)*, **343**, 687–689.

[struct_mon_prot]

STRUCT_MON_PROT_CIS

Data items in the STRUCT_MON_PROT_CIS category identify monomers that have been found to have the peptide bond in the *cis* conformation. The criterion used to select residues to be designated as containing *cis* peptide bonds is given in **_struct_mon_details.prot_cis**.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_mon_prot_cis.label_alt_id**
_struct_mon_prot_cis.label_asym_id
_struct_mon_prot_cis.label_comp_id
_struct_mon_prot_cis.label_seq_id

*Example 1 – based on PDB structure 1ACY of Ghiara, Stura, Stanfield, Profy & Wilson [Science (1994), **264**, 82–85].*

```
loop_
_struct_mon_prot_cis.label_comp_id
_struct_mon_prot_cis.label_seq_id
_struct_mon_prot_cis.label_asym_id
_struct_mon_prot_cis.label_alt_id
 PRO   8  L .
 PRO  77  L .
 PRO  95  L .
 PRO 141  L .
 #  -----   abbreviated  -----
```

_struct_mon_prot_cis.auth_asym_id

A component of the identifier for the monomer. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_mon_prot_cis.auth_comp_id

A component of the identifier for the monomer. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_mon_prot_cis.auth_seq_id

A component of the identifier for the monomer.This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

***_struct_mon_prot_cis.label_alt_id**

A component of the identifier for the monomer. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_mon_prot_cis.label_asym_id**

A component of the identifier for the monomer. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_mon_prot_cis.label_comp_id**

A component of the identifier for the monomer. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_mon_prot_cis.label_seq_id**

A component of the identifier for the monomer. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

STRUCT_NCS_DOM

Data items in the STRUCT_NCS_DOM category record information about the domains in an ensemble of domains related by one or more noncrystallographic symmetry operators. A domain need not correspond to a complete polypeptide chain; it can be composed of one or more segments in a single chain, or by segments from more than one chain.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_ncs_dom.id**

Example 1 – based on laboratory records for the collagen-like peptide, HYP-.

```
loop_
_struct_ncs_dom.id
_struct_ncs_dom.details
 d1  'Chains A, B, and C'
 d2  'Chains D, E, and F'
```

_struct_ncs_dom.details (*text*)

A description of special aspects of the structural elements that comprise a domain in an ensemble of domains related by noncrystallographic symmetry.

Example:

```
; The loop between residues 18 and 23 in this domain interacts
with a symmetry-related molecule, and thus deviates
significantly from the noncrystallographic threefold.
;
```

[struct_ncs_dom]

***_struct_ncs_dom.id** (*code*)
The value of **_struct_ncs_dom.id** must uniquely identify a record in the STRUCT_NCS_DOM list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_struct_ncs_dom_lim.dom_id,
_struct_ncs_ens_gen.dom_id_1,
_struct_ncs_ens_gen.dom_id_2. **[struct_ncs_dom]**

STRUCT_NCS_DOM_LIM

Data items in the STRUCT_NCS_DOM_LIM category identify the start and end points of polypeptide chain segments that form all or part of a domain in an ensemble of domains related by non-crystallographic symmetry.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_ncs_dom_lim.dom_id**
_struct_ncs_dom_lim.beg_label_alt_id
_struct_ncs_dom_lim.beg_label_asym_id
_struct_ncs_dom_lim.beg_label_comp_id
_struct_ncs_dom_lim.beg_label_seq_id
_struct_ncs_dom_lim.end_label_alt_id
_struct_ncs_dom_lim.end_label_asym_id
_struct_ncs_dom_lim.end_label_comp_id
_struct_ncs_dom_lim.end_label_seq_id

Example 1 – based on laboratory records for the collagen-like peptide, HYP-.

```
loop_
_struct_ncs_dom_lim.dom_id
_struct_ncs_dom_lim.beg_label_alt_id
_struct_ncs_dom_lim.beg_label_asym_id
_struct_ncs_dom_lim.beg_label_comp_id
_struct_ncs_dom_lim.beg_label_seq_id
_struct_ncs_dom_lim.end_label_alt_id
_struct_ncs_dom_lim.end_label_asym_id
_struct_ncs_dom_lim.end_label_comp_id
_struct_ncs_dom_lim.end_label_seq_id
 d1   .  A PRO  1   . A GLY  29
 d1   .  B PRO  31  . B GLY  59
 d1   .  C PRO  61  . B GLY  89
 d2   .  D PRO  91  . D GLY 119
 d2   .  E PRO 121  . E GLY 149
 d2   .  F PRO 151  . F GLY 179
```

_struct_ncs_dom_lim.beg_auth_asym_id
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_ncs_dom_lim.beg_auth_comp_id
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_ncs_dom_lim.beg_auth_seq_id
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.beg_label_alt_id**
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_ncs_dom_lim.beg_label_asym_id**
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.beg_label_comp_id**
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.beg_label_seq_id**
A component of the identifier for the monomer at which this segment of the domain begins. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.dom_id**
This data item is a pointer to **_struct_ncs_dom.id** in the STRUCT_NCS_DOM category.

_struct_ncs_dom_lim.end_auth_asym_id
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_ncs_dom_lim.end_auth_comp_id
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_ncs_dom_lim.end_auth_seq_id
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.end_label_alt_id**
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_ncs_dom_lim.end_label_asym_id**
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.end_label_comp_id**
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_ncs_dom_lim.end_label_seq_id**
A component of the identifier for the monomer at which this segment of the domain ends. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

STRUCT_NCS_ENS

Data items in the STRUCT_NCS_ENS category record information about ensembles of domains related by noncrystallographic symmetry. The point group of the ensemble when taken as a whole may be specified, as well as any special aspects of the ensemble that require description.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_ncs_ens.id**

Example 1 – based on laboratory records for the collagen-like peptide, HYP-.

```
_struct_ncs_ens.id          en1
_struct_ncs_ens.details
; The ensemble represents the pseudo-twofold symmetry
  between domains d1 and d2.
;
```

_struct_ncs_ens.details (*text*)

A description of special aspects of the ensemble.

Example:

```
; The ensemble has a slight translation between domains 1
and 4, but overall it can accurately be described as point
group 222
;
```

[struct_ncs_ens]

* **_struct_ncs_ens.id** (*code*)

The value of **_struct_ncs_ens.id** must uniquely identify a record in the STRUCT_NCS_ENS list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_struct_ncs_ens_gen.ens_id.

[struct_ncs_ens]

_struct_ncs_ens.point_group (*line*)

The point group of the ensemble of structural elements related by one or more noncrystallographic symmetry operations. The relationships need not be precise; this data item is intended to give a rough description of the noncrystallographic symmetry relationships.

Examples: '3', '422', 'non-proper'.

[struct_ncs_ens]

STRUCT_NCS_ENS_GEN

Data items in the STRUCT_NCS_ENS_GEN category list domains related by a noncrystallographic symmetry operation and identify the operator.

Category group(s): **inclusive_group**
struct_group

Category key(s): **_struct_ncs_ens_gen.ens_id**
_struct_ncs_ens_gen.dom_id_1
_struct_ncs_ens_gen.dom_id_2
_struct_ncs_ens_gen.oper_id

Example 1 – based on laboratory records for the collagen-like peptide, HYP-.

```
_struct_ncs_ens_gen.dom_id_1  d1
_struct_ncs_ens_gen.dom_id_2  d2
_struct_ncs_ens_gen.ens_id    en1
_struct_ncs_ens_gen.oper_id   ncsop1
```

* **_struct_ncs_ens_gen.dom_id_1**

The identifier for the domain that will remain unchanged by the transformation operator. This data item is a pointer to **_struct_ncs_dom.id** in the STRUCT_NCS_DOM category.

* **_struct_ncs_ens_gen.dom_id_2**

The identifier for the domain that will be transformed by application of the transformation operator. This data item is a pointer to **_struct_ncs_dom.id** in the STRUCT_NCS_DOM category.

* **_struct_ncs_ens_gen.ens_id**

This data item is a pointer to **_struct_ncs_ens.id** in the STRUCT_NCS_ENS category.

* **_struct_ncs_ens_gen.oper_id**

This data item is a pointer to **_struct_ncs_oper.id** in the STRUCT_NCS_OPER category.

STRUCT_NCS_OPER

Data items in the STRUCT_NCS_OPER category describe the noncrystallographic symmetry operations. Each operator is specified as a matrix and a subsequent translation vector. Operators need not represent proper rotations.

Category group(s): **inclusive_group**
struct_group

Category key(s): **_struct_ncs_oper.id**

Example 1 – based on laboratory records for the protein NS1.

```
_struct_ncs_oper.id             ncsop1
_struct_ncs_oper.code           given
_struct_ncs_oper.matrix[1][1]   0.247
_struct_ncs_oper.matrix[1][2]   0.935
_struct_ncs_oper.matrix[1][3]   0.256
_struct_ncs_oper.matrix[2][1]   0.929
_struct_ncs_oper.matrix[2][2]   0.153
_struct_ncs_oper.matrix[2][3]   0.337
_struct_ncs_oper.matrix[3][1]   0.276
_struct_ncs_oper.matrix[3][2]   0.321
_struct_ncs_oper.matrix[3][3]  -0.906
_struct_ncs_oper.vector[1]     -8.253
_struct_ncs_oper.vector[2]    -11.743
_struct_ncs_oper.vector[3]     -1.782
_struct_ncs_oper.details
; Matrix and translation vector for pseudo-twofold operation.
;
```

_struct_ncs_oper.code (*code*)

A code to indicate whether this operator describes a relationship between coordinates all of which are given in the data block (in which case the value of code is 'given'), or whether the operator is used to generate new coordinates from those that are given in the data block (in which case the value of code is 'generate').

The data value must be one of the following:

given	operator relates coordinates given in the data block
generate	operator generates new coordinates from those given in the data block

[struct_ncs_oper]

_struct_ncs_oper.details (*text*)

A description of special aspects of the noncrystallographic symmetry operator.

Example:

```
; The operation is given as a precise threefold rotation,
despite the fact the best rms fit between domain 1 and domain
2 yields a rotation of 119.7 degrees and a translation of
0.13 angstroms.
;
```

[struct_ncs_oper]

* **_struct_ncs_oper.id** (*code*)

The value of **_struct_ncs_oper.id** must uniquely identify a record in the STRUCT_NCS_OPER list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:

_struct_ncs_ens_gen.oper_id.

[struct_ncs_oper]

_struct_ncs_oper.matrix[1][1] (*float*)

The [1][1] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[1][2] (*float*)

The [1][2] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[1][3] (*float*)

The [1][3] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[2][1] (*float*)
The [2][1] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[2][2] (*float*)
The [2][2] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[2][3] (*float*)
The [2][3] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[3][1] (*float*)
The [3][1] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[3][2] (*float*)
The [3][2] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.matrix[3][3] (*float*)
The [3][3] element of the 3 × 3 matrix component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.vector[1] (*float*)
The [1] element of the three-element vector component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.vector[2] (*float*)
The [2] element of the three-element vector component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

_struct_ncs_oper.vector[3] (*float*)
The [3] element of the three-element vector component of a noncrystallographic symmetry operation.

[struct_ncs_oper]

STRUCT_REF

Data items in the STRUCT_REF category allow the author of a data block to relate the entities or biological units described in the data block to information archived in external databases. For references to the sequence of a polymer, the value of the data item **_struct_ref.seq_align** is used to indicate whether the correspondence between the sequence of the entity or biological unit in the data block and the sequence in the referenced database entry is 'complete' or 'partial'. If this value is 'partial', the region (or regions) of the alignment may be delimited using data items in the STRUCT_REF_SEQ category. Similarly, the value of **_struct_ref.seq_dif** is used to indicate whether the two sequences contain point differences. If the value is 'yes', the differences may be identified and annotated using data items in the STRUCT_REF_SEQ_DIF category.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_ref.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_ref.id
_struct_ref.entity_id
_struct_ref.biol_id
_struct_ref.db_name
_struct_ref.db_code
_struct_ref.seq_align
_struct_ref.seq_dif
_struct_ref.details
 seq_pdb     .  1  PDB      5HVP      .         .
; The structure of the closely related compound,
  isobutyryl-pepstatin (pepstatin A) in complex with
  rhizopuspepsin
;
 seq_genbank 1  .  GenBank  AAG30358  complete  yes  .
```

_struct_ref.biol_id
This data item is a pointer to **_struct_biol.id** in the STRUCT_BIOL category.

***_struct_ref.db_code** (*line*)
The code for this entity or biological unit or for a closely related entity or biological unit in the named database.
Examples: '1ABC', 'ABCDEF'. **[struct_ref]**

***_struct_ref.db_name** (*line*)
The name of the database containing reference information about this entity or biological unit.
Examples: 'PDB', 'CSD', 'Genbank'. **[struct_ref]**

_struct_ref.details (*text*)
A description of special aspects of the relationship between the entity or biological unit described in the data block and that in the referenced database entry.

[struct_ref]

***_struct_ref.entity_id**
This data item is a pointer to **_entity.id** in the ENTITY category.

***_struct_ref.id** (*code*)
The value of **_struct_ref.id** must uniquely identify a record in the STRUCT_REF list. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_struct_ref_seq.ref_id. **[struct_ref]**

_struct_ref.seq_align (*ucode*)
A flag to indicate the scope of the alignment between the sequence of the entity or biological unit described in the data block and that in the referenced database entry. 'complete' indicates that alignment spans the entire length of both sequences (although point differences may occur and can be annotated using the data items in the STRUCT_REF_SEQ_DIF category). 'partial' indicates a partial alignment. The region (or regions) of the alignment may be delimited using data items in the STRUCT_REF_SEQ category. This data item may also take the value '.', indicating that the reference is not to a sequence.
The data value must be one of the following:

complete	alignment is complete
partial	alignment is partial
.	reference is not to a sequence

[struct_ref]

_struct_ref.seq_dif (*ucode*)
A flag to indicate the presence ('yes') or absence ('no') of point differences between the sequence of the entity or biological unit described in the data block and that in the referenced database entry. This data item may also take the value '.', indicating that the reference is not to a sequence.

The data value must be one of the following:

no	there are no point differences
n	abbreviation for 'no'
yes	there are point difference
y	abbreviation for 'yes'
.	reference is not to a sequence

[struct_ref]

STRUCT_REF_SEQ

Data items in the STRUCT_REF_SEQ category provide a mechanism for indicating and annotating a region (or regions) of alignment between the sequence of an entity or biological unit described in the data block and the sequence in the referenced database entry.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_ref_seq.align_id**

Example 1 – based on the sequence alignment of CHER from M. xantus (36 to 288) and CHER from S. typhimurium (18 to 276).

```
_struct_ref_seq.align_id          alg1
_struct_ref_seq.ref_id            seqdb1
_struct_ref_seq.seq_align_beg     36
_struct_ref_seq.seq_align_end     288
_struct_ref_seq.db_align_beg      18
_struct_ref_seq.db_align_end      276
_struct_ref_seq.details
; The alignment contains 3 gaps larger than 2 residues
;
```

***_struct_ref_seq.align_id** (*code*)
The value of **_struct_ref_seq.align_id** must uniquely identify a record in the STRUCT_REF_SEQ list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:
_struct_ref_seq_dif.align_id. **[struct_ref_seq]**

***_struct_ref_seq.db_align_beg** (*int*)
The sequence position in the referenced database entry at which the alignment begins.

***_struct_ref_seq.db_align_end** (*int*)
The sequence position in the referenced database entry at which the alignment ends.

_struct_ref_seq.details (*text*)
A description of special aspects of the sequence alignment.
[struct_ref_seq]

***_struct_ref_seq.ref_id**
This data item is a pointer to **_struct_ref.id** in the STRUCT_REF category.

***_struct_ref_seq.seq_align_beg**
The sequence position in the entity or biological unit described in the data block at which the alignment begins. This data item is a pointer to **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category.

***_struct_ref_seq.seq_align_end**
The sequence position in the entity or biological unit described in the data block at which the alignment ends. This data item is a pointer to **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category.

STRUCT_REF_SEQ_DIF

Data items in the STRUCT_REF_SEQ_DIF category provide a mechanism for indicating and annotating point differences between the sequence of the entity or biological unit described in the data block and the sequence of the referenced database entry.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_ref_seq_dif.align_id**
_struct_ref_seq_dif.seq_num

Example 1 – based on laboratory records for CAP-DNA complex.

```
_struct_ref_seq_dif.align_id   algn2
_struct_ref_seq_dif.seq_num    181
_struct_ref_seq_dif.db_mon_id  GLU
_struct_ref_seq_dif.mon_id     PHE
_struct_ref_seq_dif.details
; A point mutation was introduced in the CAP at position 181
  substituting PHE for GLU.
;
```

***_struct_ref_seq_dif.align_id**
This data item is a pointer to **_struct_ref_seq.align_id** in the STRUCT_REF_SEQ category.

***_struct_ref_seq_dif.db_mon_id**
The monomer type found at this position in the referenced database entry. This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

_struct_ref_seq_dif.details (*text*)
A description of special aspects of the point differences between the sequence of the entity or biological unit described in the data block and that in the referenced database entry.
[struct_ref_seq_dif]

***_struct_ref_seq_dif.mon_id**
The monomer type found at this position in the sequence of the entity or biological unit described in this data block. This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

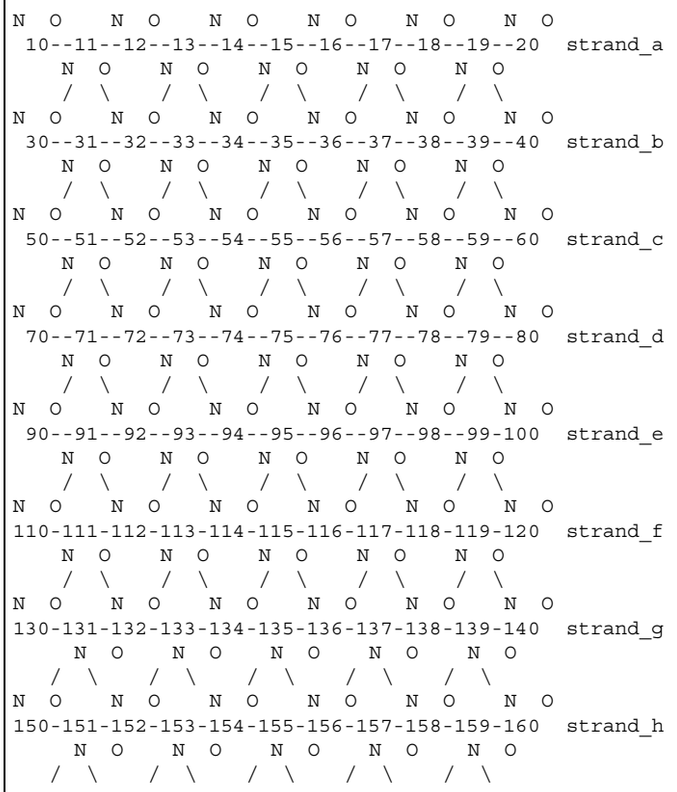
***_struct_ref_seq_dif.seq_num**
This data item is a pointer to **_entity_poly_seq.num** in the ENTITY_POLY_SEQ category.

STRUCT_SHEET

Data items in the STRUCT_SHEET category record details about the β-sheets.

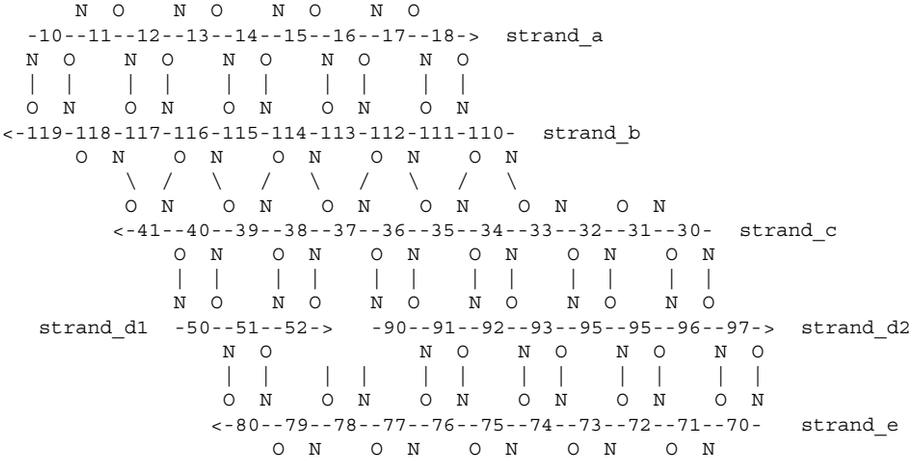
Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_sheet.id**

Example 1 - simple beta-barrel.




```
_struct_sheet.id               sheet_1
_struct_sheet.type            'beta-barrel'
_struct_sheet.number_strands   8
_struct_sheet.details          .
```

Example 2 - five stranded mixed-sense sheet with one two-piece strand.




```
_struct_sheet.id               sheet_2
_struct_sheet.type            'five stranded, mixed-sense'
_struct_sheet.number_strands   5
_struct_sheet.details         'strand_d is in two pieces'
```

_struct_sheet.details (*text*)
A description of special aspects of the β-sheet.
[struct_sheet]

* **_struct_sheet.id** (*code*)
The value of **_struct_sheet.id** must uniquely identify a record in the STRUCT_SHEET list. Note that this item need not be a number; it can be any unique identifier.
The following item(s) have an equivalent role in their respective categories:
_struct_sheet_hbond.sheet_id,
_struct_sheet_order.sheet_id,
_struct_sheet_range.sheet_id,
_struct_sheet_topology.sheet_id. **[struct_sheet]**

_struct_sheet.number_strands (*int*)
The number of strands in the sheet. If a given range of residues bulges out from the strands, it is still counted as one strand. If a strand is composed of two different regions of polypeptide, it is still counted as one strand, as long as the proper hydrogen-bonding connections are made to adjacent strands.
[struct_sheet]

_struct_sheet.type (*text*)
A simple descriptor for the type of the sheet.
Examples: 'jelly-roll', 'Rossmann fold', 'beta barrel'. **[struct_sheet]**

STRUCT_SHEET_HBOND

Data items in the STRUCT_SHEET_HBOND category record details about the hydrogen bonding between residue ranges in a β-sheet. It is necessary to treat hydrogen bonding independently of the designation of ranges, because the hydrogen bonding may begin in different places for the interactions of a given strand with the one preceding it and the one following it in the sheet.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_sheet_hbond.sheet_id**
_struct_sheet_hbond.range_id_1
_struct_sheet_hbond.range_id_2

Example 1 – simple beta-barrel.

```
loop_
_struct_sheet_hbond.sheet_id
_struct_sheet_hbond.range_id_1
_struct_sheet_hbond.range_id_2
_struct_sheet_hbond.range_1_beg_label_seq_id
_struct_sheet_hbond.range_1_beg_label_atom_id
_struct_sheet_hbond.range_2_beg_label_seq_id
_struct_sheet_hbond.range_2_beg_label_atom_id
_struct_sheet_hbond.range_1_end_label_seq_id
_struct_sheet_hbond.range_1_end_label_atom_id
_struct_sheet_hbond.range_2_end_label_seq_id
_struct_sheet_hbond.range_2_end_label_atom_id
 sheet_1  strand_a  strand_b   11  N   30  O   19  O   40  N
 sheet_1  strand_b  strand_c   31  N   50  O   39  O   60  N
 sheet_1  strand_c  strand_d   51  N   70  O   59  O   80  N
 sheet_1  strand_d  strand_e   71  N   90  O   89  O  100  N
 sheet_1  strand_e  strand_f   91  N  110  O   99  O  120  N
 sheet_1  strand_f  strand_g  111  N  130  O  119  O  140  N
 sheet_1  strand_g  strand_h  131  N  150  O  139  O  160  N
 sheet_1  strand_h  strand_a  151  N   10  O  159  O  180  N
```

Example 2 – five stranded mixed-sense sheet with one two-piece strand.

```
loop_
_struct_sheet_hbond.sheet_id
_struct_sheet_hbond.range_id_1
_struct_sheet_hbond.range_id_2
_struct_sheet_hbond.range_1_beg_label_seq_id
_struct_sheet_hbond.range_1_beg_label_atom_id
_struct_sheet_hbond.range_2_beg_label_seq_id
_struct_sheet_hbond.range_2_beg_label_atom_id
_struct_sheet_hbond.range_1_end_label_seq_id
_struct_sheet_hbond.range_1_end_label_atom_id
_struct_sheet_hbond.range_2_end_label_seq_id
_struct_sheet_hbond.range_2_end_label_atom_id
 sheet_2  strand_a   strand_b    20  N  119  O  18  O  111  N
 sheet_2  strand_b   strand_c   110  N   33  O 118  N   41  O
 sheet_2  strand_c   strand_d1   38  N   52  O  40  O   50  N
 sheet_2  strand_c   strand_d2   30  N   96  O  36  O   90  N
 sheet_2  strand_d1  strand_e    51  N   80  O  51  O   80  N
 sheet_2  strand_d2  strand_e    91  N   76  O  97  O   70  N
```

_struct_sheet_hbond.range_1_beg_auth_atom_id
A component of the identifier for the residue for the first partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_1_beg_auth_seq_id
A component of the identifier for the residue for the first partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_struct_sheet_hbond.range_1_beg_label_atom_id**
A component of the identifier for the residue for the first partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

* **_struct_sheet_hbond.range_1_beg_label_seq_id**
A component of the identifier for the residue for the first partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_1_end_auth_atom_id
A component of the identifier for the residue for the first partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_1_end_auth_seq_id
A component of the identifier for the residue for the first partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_struct_sheet_hbond.range_1_end_label_atom_id**
A component of the identifier for the residue for the first partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

* **_struct_sheet_hbond.range_1_end_label_seq_id**
A component of the identifier for the residue for the first partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_2_beg_auth_atom_id
A component of the identifier for the residue for the second partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_2_beg_auth_seq_id
A component of the identifier for the residue for the second partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

*__struct_sheet_hbond.range_2_beg_label_atom_id**
A component of the identifier for the residue for the second partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

*__struct_sheet_hbond.range_2_beg_label_seq_id**
A component of the identifier for the residue for the second partner of the first hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_2_end_auth_atom_id
A component of the identifier for the residue for the second partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_sheet_hbond.range_2_end_auth_seq_id
A component of the identifier for the residue for the second partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

*__struct_sheet_hbond.range_2_end_label_atom_id**
A component of the identifier for the residue for the second partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_atom_id** in the ATOM_SITE category.

*__struct_sheet_hbond.range_2_end_label_seq_id**
A component of the identifier for the residue for the second partner of the last hydrogen bond between two residue ranges in a sheet. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

*__struct_sheet_hbond.range_id_1**
This data item is a pointer to **_struct_sheet_range.id** in the STRUCT_SHEET_RANGE category.

*__struct_sheet_hbond.range_id_2**
This data item is a pointer to **_struct_sheet_range.id** in the STRUCT_SHEET_RANGE category.

*__struct_sheet_hbond.sheet_id**
This data item is a pointer to **_struct_sheet.id** in the STRUCT_SHEET category.

STRUCT_SHEET_ORDER

Data items in the STRUCT_SHEET_ORDER category record details about the order of the residue ranges that form a β-sheet. All order links are pairwise and the specified pairs are assumed to be adjacent to one another in the sheet. These data items are an alternative to the STRUCT_SHEET_TOPOLOGY data items and they allow all manner of sheets to be described.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_sheet_order.sheet_id**
_struct_sheet_order.range_id_1
_struct_sheet_order.range_id_2

Example 1 – simple beta-barrel.

```
loop_
_struct_sheet_order.sheet_id
_struct_sheet_order.range_id_1
_struct_sheet_order.range_id_2
_struct_sheet_order.offset
_struct_sheet_order.sense
 sheet_1  strand_a  strand_b  +1  parallel
 sheet_1  strand_b  strand_c  +1  parallel
 sheet_1  strand_c  strand_d  +1  parallel
 sheet_1  strand_d  strand_e  +1  parallel
 sheet_1  strand_e  strand_f  +1  parallel
 sheet_1  strand_f  strand_g  +1  parallel
 sheet_1  strand_g  strand_h  +1  parallel
 sheet_1  strand_h  strand_a  +1  parallel
```

Example 2 – five stranded mixed-sense sheet with one two-piece strand.

```
loop_
_struct_sheet_order.sheet_id
_struct_sheet_order.range_id_1
_struct_sheet_order.range_id_2
_struct_sheet_order.offset
_struct_sheet_order.sense
 sheet_2  strand_a   strand_b    +1 anti-parallel
 sheet_2  strand_b   strand_c    +1 parallel
 sheet_2  strand_c   strand_d1   +1 anti-parallel
 sheet_2  strand_c   strand_d2   +1 anti-parallel
 sheet_2  strand_d1  strand_e    +1 anti-parallel
 sheet_2  strand_d2  strand_e    +1 anti-parallel
```

_struct_sheet_order.offset (*int*)
Designates the relative position in the sheet, plus or minus, of the second residue range to the first.
[struct_sheet_order]

*__struct_sheet_order.range_id_1**
This data item is a pointer to **_struct_sheet_range.id** in the STRUCT_SHEET_RANGE category.

*__struct_sheet_order.range_id_2**
This data item is a pointer to **_struct_sheet_range.id** in the STRUCT_SHEET_RANGE category.

_struct_sheet_order.sense (*ucode*)
A flag to indicate whether the two designated residue ranges are parallel or antiparallel to one another.
The data value must be one of the following:
parallel
anti-parallel
[struct_sheet_order]

*__struct_sheet_order.sheet_id**
This data item is a pointer to **_struct_sheet.id** in the STRUCT_SHEET category.

STRUCT_SHEET_RANGE

Data items in the STRUCT_SHEET_RANGE category record details about the residue ranges that form a β-sheet. Residues are included in a range if they made β-sheet-type hydrogen-bonding interactions with at least one adjacent strand and if there are at least two residues in the range.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_sheet_range.sheet_id**
_struct_sheet_range.id

Example 1 – simple beta-barrel.

```
loop_
_struct_sheet_range.sheet_id
_struct_sheet_range.id
_struct_sheet_range.beg_label_comp_id
_struct_sheet_range.beg_label_asym_id
_struct_sheet_range.beg_label_seq_id
_struct_sheet_range.end_label_comp_id
_struct_sheet_range.end_label_asym_id
_struct_sheet_range.end_label_seq_id
_struct_sheet_range.symmetry
 sheet_1  strand_a  ala  A   20  ala  A   30  1_555
 sheet_1  strand_b  ala  A   40  ala  A   50  1_555
 sheet_1  strand_c  ala  A   60  ala  A   70  1_555
 sheet_1  strand_d  ala  A   80  ala  A   90  1_555
 sheet_1  strand_e  ala  A  100  ala  A  110  1_555
 sheet_1  strand_f  ala  A  120  ala  A  130  1_555
 sheet_1  strand_g  ala  A  140  ala  A  150  1_555
 sheet_1  strand_h  ala  A  160  ala  A  170  1_555
```

Example 2 – five stranded mixed-sense sheet with one two-piece strand.

```
loop_
_struct_sheet_range.sheet_id
_struct_sheet_range.id
_struct_sheet_range.beg_label_comp_id
_struct_sheet_range.beg_label_asym_id
_struct_sheet_range.beg_label_seq_id
_struct_sheet_range.end_label_comp_id
_struct_sheet_range.end_label_asym_id
_struct_sheet_range.end_label_seq_id
_struct_sheet_range.symmetry
 sheet_2  strand_a   ala  A  10  ala  A  18  1_555
 sheet_2  strand_b   ala  A 110  ala  A 119  1_555
 sheet_2  strand_c   ala  A  30  ala  A  41  1_555
 sheet_2  strand_d1  ala  A  50  ala  A  52  1_555
 sheet_2  strand_d2  ala  A  90  ala  A  97  1_555
 sheet_2  strand_e   ala  A  70  ala  A  80  1_555
```

_struct_sheet_range.beg_auth_asym_id
A component of the identifier for the residue at which the β-sheet range begins. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_sheet_range.beg_auth_comp_id
A component of the identifier for the residue at which the β-sheet range begins. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_sheet_range.beg_auth_seq_id
A component of the identifier for the residue at which the β-sheet range begins. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_struct_sheet_range.beg_label_asym_id**
A component of the identifier for the residue at which the β-sheet range begins. This data item is a pointer to **_struct_asym.id** in the STRUCT_ASYM category.

* **_struct_sheet_range.beg_label_comp_id**
A component of the identifier for the residue at which the β-sheet range begins. This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

* **_struct_sheet_range.beg_label_seq_id**
A component of the identifier for the residue at which the β-sheet range begins. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

_struct_sheet_range.end_auth_asym_id
A component of the identifier for the residue at which the β-sheet range ends. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_sheet_range.end_auth_comp_id
A component of the identifier for the residue at which the β-sheet range ends. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_sheet_range.end_auth_seq_id
A component of the identifier for the residue at which the β-sheet range ends. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

* **_struct_sheet_range.end_label_asym_id**
A component of the identifier for the residue at which the β-sheet range ends. This data item is a pointer to **_struct_asym.id** in the STRUCT_ASYM category.

* **_struct_sheet_range.end_label_comp_id**
A component of the identifier for the residue at which the β-sheet range ends. This data item is a pointer to **_chem_comp.id** in the CHEM_COMP category.

* **_struct_sheet_range.end_label_seq_id**
A component of the identifier for the residue at which the β-sheet range ends. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

* **_struct_sheet_range.id** (*code*)
The value of **_struct_sheet_range.id** must uniquely identify a range in a given sheet in the STRUCT_SHEET_RANGE list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:
_struct_sheet_hbond.range_id_1,
_struct_sheet_hbond.range_id_2,
_struct_sheet_order.range_id_1,
_struct_sheet_order.range_id_2,
_struct_sheet_topology.range_id_1,
_struct_sheet_topology.range_id_2. **[struct_sheet_range]**

* **_struct_sheet_range.sheet_id**
This data item is a pointer to **_struct_sheet.id** in the STRUCT_SHEET category.

_struct_sheet_range.symmetry (*symop*)
Describes the symmetry operation that should be applied to the residues delimited by the start and end designators in order to generate the appropriate strand in this sheet.
[struct_sheet_range]

STRUCT_SHEET_TOPOLOGY

Data items in the STRUCT_SHEET_TOPOLOGY category record details about the topology of the residue ranges that form a β-sheet. All topology links are pairwise and the specified pairs are assumed to be successive in the amino-acid sequence. These data items are useful in describing various simple and complex folds, but they become inadequate when the strands in the sheet come from more than one chain. The STRUCT_SHEET_ORDER data items can be used to describe single- and multiple-chain-containing sheets.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_sheet_topology.sheet_id**
_struct_sheet_topology.range_id_1
_struct_sheet_topology.range_id_2

Example 1 – simple beta-barrel.

```
loop_
_struct_sheet_topology.sheet_id
_struct_sheet_topology.range_id_1
_struct_sheet_topology.range_id_2
_struct_sheet_topology.offset
_struct_sheet_topology.sense
 sheet_1  strand_a  strand_b  +1  parallel
 sheet_1  strand_b  strand_c  +1  parallel
 sheet_1  strand_c  strand_d  +1  parallel
 sheet_1  strand_d  strand_e  +1  parallel
 sheet_1  strand_e  strand_f  +1  parallel
 sheet_1  strand_f  strand_g  +1  parallel
 sheet_1  strand_g  strand_h  +1  parallel
 sheet_1  strand_h  strand_a  +1  parallel
```

Example 2 – five stranded mixed-sense sheet with one two-piece strand.

```
loop_
_struct_sheet_topology.sheet_id
_struct_sheet_topology.range_id_1
_struct_sheet_topology.range_id_2
_struct_sheet_topology.offset
_struct_sheet_topology.sense
 sheet_2  strand_a   strand_c    +2 anti-parallel
 sheet_2  strand_c   strand_d1   +1 anti-parallel
 sheet_2  strand_d1  strand_e    +1 anti-parallel
 sheet_2  strand_e   strand_d2   -1 anti-parallel
 sheet_2  strand_d2  strand_b    -2 anti-parallel
```

_struct_sheet_topology.offset (*int*)
Designates the relative position in the sheet, plus or minus, of the second residue range to the first.

[struct_sheet_topology]

***_struct_sheet_topology.range_id_1**
This data item is a pointer to **_struct_sheet_range.id** in the STRUCT_SHEET_RANGE category.

***_struct_sheet_topology.range_id_2**
This data item is a pointer to **_struct_sheet_range.id** in the STRUCT_SHEET_RANGE category.

_struct_sheet_topology.sense (*ucode*)
A flag to indicate whether the two designated residue ranges are parallel or antiparallel to one another.

The data value must be one of the following:

parallel
anti-parallel

[struct_sheet_topology]

***_struct_sheet_topology.sheet_id**
This data item is a pointer to **_struct_sheet.id** in the STRUCT_SHEET category.

STRUCT_SITE

Data items in the STRUCT_SITE category record details about portions of the structure that contribute to structurally relevant sites (*e.g.* active sites, substrate-binding subsites, metal-coordination sites).

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_site.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_site.id
_struct_site.details
 'P2 site C'
; residues with a contact < 3.7 \%A to an atom in the P2
  moiety of the inhibitor in the conformation with
     _struct_asym.id = C
;
 'P2 site D'
; residues with a contact < 3.7 \%A to an atom in the P1
  moiety of the inhibitor in the conformation with
     _struct_asym.id = D)
;
```

_struct_site.details (*text*)
A description of special aspects of the site.

[struct_site]

***_struct_site.id** (*line*)
The value of **_struct_site.id** must uniquely identify a record in the STRUCT_SITE list. Note that this item need not be a number; it can be any unique identifier.

The following item(s) have an equivalent role in their respective categories:
_struct_site_gen.site_id,
_struct_site_keywords.site_id,
_struct_site_view.site_id. **[struct_site]**

STRUCT_SITE_GEN

Data items in the STRUCT_SITE_GEN category record details about the generation of portions of the structure that contribute to structurally relevant sites.

Category group(s): **inclusive_group**
struct_group
Category key(s): **_struct_site_gen.id**
_struct_site_gen.site_id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_site_gen.id
_struct_site_gen.site_id
_struct_site_gen.label_comp_id
_struct_site_gen.label_asym_id
_struct_site_gen.label_seq_id
_struct_site_gen.symmetry
_struct_site_gen.details
  1  1  VAL  A   32  1_555  .
  2  1  ILE  A   47  1_555  .
  3  1  VAL  A   82  1_555  .
  4  1  ILE  A   84  1_555  .
  5  2  VAL  B  232  1_555  .
  6  2  ILE  B  247  1_555  .
  7  2  VAL  B  282  1_555  .
  8  2  ILE  B  284  1_555  .
```

_struct_site_gen.auth_asym_id
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_asym_id** in the ATOM_SITE category.

_struct_site_gen.auth_atom_id
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_atom_id** in the ATOM_SITE category.

_struct_site_gen.auth_comp_id
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_comp_id** in the ATOM_SITE category.

_struct_site_gen.auth_seq_id
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.auth_seq_id** in the ATOM_SITE category.

_struct_site_gen.details (*text*)
A description of special aspects of the symmetry generation of this portion of the structural site.

Example:
```
; The zinc atom lies on a special position; application of
symmetry elements to generate  the insulin hexamer will
generate excess zinc atoms, which must be removed by hand.
;
```
[struct_site_gen]

***_struct_site_gen.id** (*line*)
The value of **_struct_site_gen.id** must uniquely identify a record in the STRUCT_SITE_GEN list. Note that this item need not be a number; it can be any unique identifier.

[struct_site_gen]

***_struct_site_gen.label_alt_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_sites_alt.id** in the ATOM_SITES_ALT category.

***_struct_site_gen.label_asym_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.label_asym_id** in the ATOM_SITE category.

***_struct_site_gen.label_atom_id**
A component of the identifier for participants in the site. This data item is a pointer to **_chem_comp_atom.atom_id** in the CHEM_COMP_ATOM category.

***_struct_site_gen.label_comp_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.label_comp_id** in the ATOM_SITE category.

***_struct_site_gen.label_seq_id**
A component of the identifier for participants in the site. This data item is a pointer to **_atom_site.label_seq_id** in the ATOM_SITE category.

***_struct_site_gen.site_id**
This data item is a pointer to **_struct_site.id** in the STRUCT_SITE category.

_struct_site_gen.symmetry (*symop*)
Describes the symmetry operation that should be applied to the atom set specified by **_struct_site_gen.label*** to generate a portion of the site.

Examples: '`.`' (no symmetry or translation to site), '`4`' (4th symmetry operation applied), '`7_645`' (7th symmetry position: $+a$ on x, $-b$ on y). **[struct_site_gen]**

STRUCT_SITE_KEYWORDS

Data items in the STRUCT_SITE_KEYWORDS category record keywords describing the site.

Category group(s): **inclusive_group**, **struct_group**
Category key(s): **_struct_site_keywords.site_id**, **_struct_site_keywords.text**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_struct_site_keywords.site_id
_struct_site_keywords.text
 'P2 site C'  'binding site'
 'P2 site C'  'binding pocket'
 'P2 site C'  'P2 site'
 'P2 site C'  'P2 pocket'
 'P2 site D'  'binding site'
 'P2 site D'  'binding pocket'
 'P2 site D'  'P2 site'
 'P2 site D'  'P2 pocket'
```

***_struct_site_keywords.site_id**
This data item is a pointer to **_struct_site.id** in the STRUCT_SITE category.

***_struct_site_keywords.text** (*text*)
Keywords describing this site.

Examples: '`active site`', '`binding pocket`', '`Ca coordination`'.
[struct_site_keywords]

STRUCT_SITE_VIEW

Data items in the STRUCT_SITE_VIEW category record details about how to draw and annotate an informative view of the site.

Category group(s): **inclusive_group**, **struct_group**
Category key(s): **_struct_site_view.id**

*Example 1 – based on NDB structure GDL001 by Coll, Aymami, Van Der Marel, Van Boom, Rich & Wang [Biochemistry (1989), **28**, 310–320].*

```
_struct_site_view.id                    1
_struct_site_view.rot_matrix[1][1]     0.132
_struct_site_view.rot_matrix[1][2]     0.922
_struct_site_view.rot_matrix[1][3]    -0.363
_struct_site_view.rot_matrix[2][1]     0.131
_struct_site_view.rot_matrix[2][2]    -0.380
_struct_site_view.rot_matrix[2][3]    -0.916
_struct_site_view.rot_matrix[3][1]    -0.982
_struct_site_view.rot_matrix[3][2]     0.073
_struct_site_view.rot_matrix[3][3]    -0.172
_struct_site_view.details
; This view highlights the site of ATAT-Netropsin
  interaction.
;
```

_struct_site_view.details (*text*)
A description of special aspects of this view of the site. This data item can be used as a figure legend.

Example:
```
; The active site has been oriented with the specificity
pocket on the right and the active site machinery on the left.
;
```
[struct_site_view]

***_struct_site_view.id** (*line*)
The value of **_struct_site_view.id** must uniquely identify a record in the STRUCT_SITE_VIEW list. Note that this item need not be a number; it can be any unique identifier.

Examples: '`Figure 1`', '`unliganded enzyme`', '`view down enzyme active site`'. **[struct_site_view]**

_struct_site_view.rot_matrix[1][1] (*float*)
The [1][1] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[1][2] (*float*)
The [1][2] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[1][3] (*float*)
The [1][3] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[2][1] (*float*)
The [2][1] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[2][2] (*float*)
The [2][2] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[2][3] (*float*)
The [2][3] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[3][1] (*float*)
The [3][1] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[3][2] (*float*)
The [3][2] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

_struct_site_view.rot_matrix[3][3] (*float*)
The [3][3] element of the matrix used to rotate the subset of the Cartesian coordinates in the ATOM_SITE category identified in the STRUCT_SITE_GEN category to an orientation useful for visualizing the site. The conventions used in the rotation are described in **_struct_site_view.details**.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}_{\text{reoriented Cartesian}} = \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Cartesian}} .$$

[struct_site_view]

***_struct_site_view.site_id**
This data item is a pointer to **_struct_site.id** in the STRUCT_SITE category.

SYMMETRY

Data items in the SYMMETRY category record details about the space-group symmetry.

Category group(s): **inclusive_group**
symmetry_group
Category key(s): **_symmetry.entry_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_symmetry.entry_id                      '5HVP'
_symmetry.cell_setting                  orthorhombic
_symmetry.Int_Tables_number             18
_symmetry.space_group_name_H-M          'P 21 21 2'
```

_symmetry.cell_setting (*ucode*)
_symmetry_cell_setting (*cif_core.dic 2.0.1*)
The cell settings for this space-group symmetry.
The data value must be one of the following:
triclinic
monoclinic
orthorhombic
tetragonal
rhombohedral
trigonal
hexagonal
cubic
[symmetry]

*__symmetry.entry_id**
This data item is a pointer to **_entry.id** in the ENTRY category.

_symmetry.Int_Tables_number (*int*)
_symmetry_Int_Tables_number (*cif_core.dic 2.0.1*)
Space-group number from *International Tables for Crystallography* Vol. A (2002).
[symmetry]

_symmetry.space_group_name_H-M (*line*)
_symmetry_space_group_name_H-M (*cif_core.dic 2.0.1*)
Hermann–Mauguin space-group symbol. Note that the Hermann–Mauguin symbol does not necessarily contain complete information about the symmetry and the space-group origin. If used, always supply the *full* symbol from *International Tables for Crystallography* Vol. A (2002) and indicate the origin and the setting if it is not implicit. If there is any doubt that the equivalent positions can be uniquely deduced from this symbol, specify the **_symmetry_equiv.pos_as_xyz** or **_symmetry.space_group_name_Hall** data items as well. Leave spaces between symbols referring to different axes.
Examples: 'P 1 21/m 1', 'P 2/n 2/n 2/n (origin at -1)', 'R -3 2/m'.
[symmetry]

_symmetry.space_group_name_Hall (*line*)
_symmetry_space_group_name_Hall (*cif_core.dic 2.0.1*)
Space-group symbol as described by Hall (1981). This symbol gives the space-group setting explicitly. Leave spaces between the separate components of the symbol.

Reference: Hall, S. R. (1981). *Acta Cryst.* A**37**, 517–525; erratum (1981), A**37**, 921.
Examples: '-P 2ac 2n', '-R 3 2"', 'P 61 2 2 (0 0 -1)'. **[symmetry]**

SYMMETRY_EQUIV

Data items in the SYMMETRY_EQUIV category list the symmetry-equivalent positions for the space group.
Category group(s): **inclusive_group**
symmetry_group
Category key(s): **_symmetry_equiv.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_symmetry_equiv.id
_symmetry_equiv.pos_as_xyz
  1  '+x,+y,+z'
  2  '-x,-y,z'
  3  '1/2+x,1/2-y,-z'
  4  '1/2-x,1/2+y,-z'
```

*__symmetry_equiv.id** (*code*)
_symmetry_equiv_pos_site_id (*cif_core.dic 2.0.1*)
The value of **_symmetry_equiv.id** must uniquely identify a record in the SYMMETRY_EQUIV category. Note that this item need not be a number; it can be any unique identifier.
[symmetry_equiv]

_symmetry_equiv.pos_as_xyz (*line*)
_symmetry_equiv_pos_as_xyz (*cif_core.dic 2.0.1*)
Symmetry-equivalent position in the '*xyz*' representation. Except for the space group *P*1, these data will be repeated in a loop. The format of the data item is as per *International Tables for Crystallography* Vol. A (2002). All equivalent positions should be entered, including those for lattice centring and a centre of symmetry, if present.
Example: '-y+x,-y,1/3+z'. **[symmetry_equiv]**

VALENCE_PARAM

Data items in the VALENCE_PARAM category define the parameters used for calculating bond valences from bond lengths. In addition to the parameters, a pointer is given to the reference (in VALENCE_REF) from which the bond-valence parameters were taken.
Category key(s): **_valence_param.atom_1**
_valence_param.atom_1_valence
_valence_param.atom_2
_valence_param.atom_2_valence

Example 1 – a bond-valence parameter list with accompanying references.

```
loop_
_valence_param.atom_1
_valence_param.atom_1_valence
_valence_param.atom_2
_valence_param.atom_2_valence
_valence_param.Ro
_valence_param.B
_valence_param.ref_id
_valence_param.details
  Cu 2 O -2 1.679 0.37 a .
  Cu 2 O -2 1.649 0.37 j .
  Cu 2 N -3 1.64  0.37 m '2-coordinate N'
  Cu 2 N -3 1.76  0.37 m '3-coordinate N'
loop_
_valence_ref.id
_valence_ref.reference
  a  'Brown & Altermatt (1985), Acta Cryst. B41, 244-247'
  j  'Liu & Thorp (1993), Inorg. Chem. 32, 4102-4205'
  m  'See, Krause & Strub (1998), Inorg. Chem. 37, 5369-5375'
```

*__valence_param.atom_1** (*code*)
_valence_param_atom_1 (*cif_core.dic 2.3*)
The element symbol of the first atom forming the bond whose bond-valence parameters are given in this category.
[valence_param]

*__valence_param.atom_1_valence** (*int*)
_valence_param_atom_1_valence (*cif_core.dic 2.3*)
The valence (formal charge) of the first atom whose bond-valence parameters are given in this category.
[valence_param]

*__valence_param.atom_2** (*code*)
_valence_param_atom_2 (*cif_core.dic 2.3*)
The element symbol of the second atom forming the bond whose bond-valence parameters are given in this category.
[valence_param]

*`_valence_param.atom_2_valence` (*int*)
`_valence_param_atom_2_valence` (*cif_core.dic 2.3*)
The valence (formal charge) of the second atom whose bond-valence parameters are given in this category.

`[valence_param]`

`_valence_param.B` (*float*)
`_valence_param_B` (*cif_core.dic 2.3*)
The bond-valence parameter B used in the expression

$$s = \exp[(R_o - R)/B],$$

where s is the valence of a bond of length R.

`[valence_param]`

`_valence_param.details` (*text*)
`_valence_param_details` (*cif_core.dic 2.3*)
Details of or comments on the bond-valence parameters.

`[valence_param]`

`_valence_param.id` (*code*)
`_valence_param_id` (*cif_core.dic 2.3*)
An identifier for the valence parameters of a bond between the given atoms.

`[valence_param]`

`_valence_param.ref_id` (*code*)
`_valence_param_ref_id` (*cif_core.dic 2.3*)
An identifier which links to the reference to the source from which the bond-valence parameters are taken. A child of `_valence_ref.id` which it must match.

`[valence_param]`

`_valence_param.Ro` (*float*)
`_valence_param_Ro` (*cif_core.dic 2.3*)
The bond-valence parameter R_o used in the expression

$$s = \exp[(R_o - R)/B],$$

where s is the valence of a bond of length R.

`[valence_param]`

VALENCE_REF

Data items in the VALENCE_REF category list the references from which the bond-valence parameters have been taken.

Category key(s): `_valence_ref.id`

*`_valence_ref.id` (*code*)
`_valence_ref_id` (*cif_core.dic 2.3*)
An identifier for items in this category. Parent of `_valence_param.ref_id`, which must have the same value.

`[valence_ref]`

`_valence_ref.reference` (*text*)
`_valence_ref_reference` (*cif_core.dic 2.3*)
Literature reference from which the valence parameters identified by `_valence_param.id` were taken.

`[valence_ref]`

4.6. Image dictionary (imgCIF)

BY A. P. HAMMERSLEY, H. J. BERNSTEIN AND J. D. WESTBROOK

This is version 1.3.2 of the image CIF dictionary (imgCIF) and crystallographic binary file dictionary (CBF) extending the macromolecular CIF dictionary (Chapter 4.5). Use of the dictionary is described in Chapter 3.7. See also Chapter 2.3 for a description of the CBF format and Chapter 5.6 for discussion of a software library for manipulating image data.

There are three category groups in this dictionary: array_data_group contains categories that describe array data; axis_group contains categories that describe axes; and diffrn_group contains categories that describe details of the diffraction experiment.

ARRAY_DATA

Data items in the ARRAY_DATA category are the containers for the array data items described in the category ARRAY_STRUCTURE.

Category group(s): **inclusive_group**
array_data_group
Category key(s): **_array_data.array_id**
_array_data.binary_id

Example 1.

This example shows two binary data blocks. The first one was compressed by the CBF_CANONICAL compression algorithm and is presented as hexadecimal data. The first character 'H' on the data lines means hexadecimal. It could have been 'O' for octal or 'D' for decimal. The second character on the line shows the number of bytes in each word (in this case '4'), which then requires eight hexadecimal digits per word. The third character gives the order of octets within a word, in this case '<' for the ordering 4321 (*i.e.* 'big-endian'). Alternatively, the character '>' could have been used for the ordering 1234 (*i.e.* 'little-endian'). The block has a 'message digest' to check the integrity of the data. The second block is similar, but uses CBF_PACKED compression and BASE64 encoding. Note that the size and the digest are different.

```
loop_
_array_data.array_id
_array_data.binary_id
_array_data.data
image_1 1
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
conversions="x-CBF_CANONICAL"
Content-Transfer-Encoding: X-BASE16
X-Binary-Size: 3927126
X-Binary-ID: 1
Content-MD5: u2sTJEovAHkmkDjPi+gWsg==

# Hexadecimal encoding, byte 0, byte order ...21
#
H4< 0050B810 00000000 00000000 00000000 000F423F 00000000 ...
....
--CIF-BINARY-FORMAT-SECTION----
;
```

```
image_2 2
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
conversions="x-CBF-PACKED"
Content-Transfer-Encoding: BASE64
X-Binary-Size: 3745758
X-Binary-ID: 2
Content-MD5: 1zsJjWPfol2GYl2V+QSXrw==

ELhQAAAAAAAA...
...
--CIF-BINARY-FORMAT-SECTION----
;
```

***_array_data.array_id** (*code*)

This item is a pointer to **_array_structure.id** in the ARRAY_STRUCTURE category.

[array_data]

(*)**_array_data.binary_id** (*int*)

This item is an integer identifier which, along with **_array_data.array_id**, should uniquely identify the particular block of array data. If **_array_data.binary_id** is not explicitly given, it defaults to 1. The value of **_array_data.binary_id** distinguishes among multiple sets of data with the same array structure. If the MIME header of the data array specifies a value for X-Binary-ID, the value of **_array_data.binary_id** should be equal to the value given for X-Binary-ID.

The following item(s) have an equivalent role in their respective categories:

_diffrn_data_frame.binary_id,
_array_intensities.binary_id.

The permitted range is $[1, \infty)$. Where no value is given, the assumed value is '1'.

[array_data]

***_array_data.data** (*binary*)

The value of **_array_data.data** contains the array data encapsulated in a STAR string. The representation used is a variant on the Multipurpose Internet Mail Extensions (MIME) specified in RFC 2045–2049 by N. Freed *et al.* The boundary delimiter used in writing an imgCIF or CBF is **--CIF-BINARY-FORMAT-SECTION--** (including the required initial **--**). The Content-Type may be any of the discrete types permitted in RFC 2045; 'application/octet-stream' is recommended. If an octet stream was compressed, the compression should be specified by the parameter **conversions="x-CBF_PACKED"** or the parameter **conversions="x-CBF_CANONICAL"**. The Content-Transfer-Encoding may be 'BASE64', 'Quoted-Printable', 'X-BASE8', 'X-BASE10' or 'X-BASE16' for an imgCIF or 'BINARY' for a CBF. The octal, decimal and hexadecimal transfer encodings are for convenience in debugging and are not recommended for archiving and data interchange. In an imgCIF file, the encoded binary data begin after the empty line terminating the header. In a CBF, the raw binary data begin after an empty line terminating the header and after the sequence

Octet	Hexadecimal	Decimal	Purpose
0	0C	12	Ctrl-L: page break
1	1A	26	Ctrl-Z: stop listings, MS-DOS
2	04	04	Ctrl-D: stop listings, UNIX
3	D5	213	binary section begins

None of these octets are included in the calculation of the message size or in the calculation of the message digest. The X-Binary-Size

Affiliations: ANDREW P. HAMMERSLEY, ESRF/EMBL Grenoble, 6 rue Jules Horowitz, France; HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA; JOHN D. WESTBROOK, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA.

header specifies the size of the equivalent binary data in octets. If compression was used, this size is the size after compression, including any book-keeping fields. An adjustment is made for the deprecated binary formats in which eight bytes of binary header are used for the compression type. In this case, the eight bytes used for the compression type are subtracted from the size, so that the same size will be reported if the compression type is supplied in the MIME header. Use of the MIME header is the recommended way to supply the compression type. In general, no portion of the binary header is included in the calculation of the size. The X-Binary-Element-Type header specifies the type of binary data in the octets, using the same descriptive phrases as in **_array_structure.encoding_type**. The default value is 'unsigned 32-bit integer'. An MD5 message digest may, optionally, be used. The *RSA Data Security, Inc. MD5 Message-Digest Algorithm* should be used. No portion of the header is included in the calculation of the message digest. If the Transfer Encoding is 'X-BASE8', 'X-BASE10' or 'X-BASE16', the data are presented as octal, decimal or hexadecimal data organized into lines or words. Each word is created by composing octets of data in fixed groups of 2, 3, 4, 6 or 8 octets, either in the order . . .4321 ('big-endian') or 1234. . . ('little-endian'). If there are fewer than the specified number of octets to fill the last word, then the missing octets are presented as '==' for each missing octet. Exactly two equal signs are used for each missing octet even for octal and decimal encoding. The format of lines is

```
rnd xxxxxx xxxxxx xxxxxx
```

where *r* is 'H', 'O' or 'D' for hexadecimal, octal or decimal, *n* is the number of octets per word and *d* is '<' or '>' for the '. . .4321' and '1234. . .' octet orderings, respectively. The '==' padding for the last word should be on the appropriate side to correspond to the missing octets, *e.g.*

```
H4< FFFFFFFF FFFFFFFF 07FFFFFF ====0000
```

or

```
H3> FF0700 00====
```

For these hexadecimal, octal and decimal formats only, comments beginning with '#' are permitted to improve readability.

BASE64 encoding follows MIME conventions. Octets are in groups of three: c1, c2, c3. The resulting 24 bits are broken into four six-bit quantities, starting with the high-order six bits (c1 $\gg$ 2) of the first octet, then the low-order two bits of the first octet followed by the high-order four bits of the second octet [(c1 & 3) $\ll$ 4 | (c2 $\gg$ 4)], then the bottom four bits of the second octet followed by the high-order two bits of the last octet [(c2 & 15) $\ll$ 2 | (c3 $\gg$ 6)], then the bottom six bits of the last octet (c3 & 63). Each of these four quantities is translated into an ASCII character using the mapping

```
1         2         3         4
01234567890123456789012345678901234567890
|         |         |         |
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn

5         6
012345678901234567890123
|         |         |
opqrstuvwxyz0123456789+/
```

with short groups of octets padded on the right with one '=' if c3 is missing, and with '==' if both c2 and c3 are missing.

QUOTED-PRINTABLE encoding also follows MIME conventions, copying octets without translation if their ASCII values are 32. . .38, 42, 48. . .57, 59, 60, 62, 64. . .126 and the octet is not a ';' in column 1. All other characters are translated to =*nn*, where *nn* is the hexadecimal encoding of the octet. All lines are 'wrapped' with a terminating = (*i.e.* the MIME conventions for an implicit line terminator are never used).

[array_data]

ARRAY_ELEMENT_SIZE

Data items in the ARRAY_ELEMENT_SIZE category record the physical size of array elements along each array dimension.

Category group(s): **inclusive_group**
array_data_group
Category key(s): **_array_element_size.array_id**
_array_element_size.index

Example 1 – a regular 2D array with a uniform element dimension of 1220 nm.

```
loop_
 _array_element_size.array_id
 _array_element_size.index
 _array_element_size.size
  image_1   1    1.22e-6
  image_1   2    1.22e-6
```

***_array_element_size.array_id** (*code*)
This item is a pointer to **_array_structure.id** in the ARRAY_STRUCTURE category.

[array_element_size]

***_array_element_size.index** (*code*)
This item is a pointer to **_array_structure_list.index** in the ARRAY_STRUCTURE_LIST category.

[array_element_size]

***_array_element_size.size** (*float*)
The size in metres of an image element in this dimension. This supposes that the elements are arranged on a regular grid.
The permitted range is $[0.0, \infty)$. **[array_element_size]**

ARRAY_INTENSITIES

Data items in the ARRAY_INTENSITIES category record the information required to recover the intensity data from the set of data values stored in the ARRAY_DATA category. The detector may have a complex relationship between the raw intensity values and the number of incident photons. In most cases, the number stored in the final array will have a simple linear relationship to the actual number of incident photons, given by **_array_intensities.gain**. If raw, uncorrected values are presented (*e.g.* for calibration experiments), the value of **_array_intensities.linearity** will be 'raw' and **_array_intensities.gain** will not be used.

Category group(s): **inclusive_group**
array_data_group
Category key(s): **_array_intensities.array_id**
_array_intensities.binary_id

Example 1.

```
loop_
  _array_intensities.array_id
  _array_intensities.linearity
  _array_intensities.gain
  _array_intensities.overload
  _array_intensities.undefined_value
   image_1   linear  1.2    655535   0
```

***_array_intensities.array_id** (*code*)
This item is a pointer to **_array_structure.id** in the ARRAY_STRUCTURE category.

[array_intensities]

(*)**_array_intensities.binary_id** (*int*)
This item is a pointer to **_array_data.binary_id** in the ARRAY_DATA category.

[array_intensities]

***_array_intensities.gain** (*float*)
Detector 'gain'. The factor by which linearized intensity count values should be divided to produce true photon counts.
The permitted range is $[0.0, \infty)$.
Related item: **_array_intensities.gain_esd** (associated value).

[array_intensities]

***_array_intensities.gain_esd** (*float*)

The estimated standard deviation in detector 'gain'.

The permitted range is $[0.0, \infty)$.

Related item: **_array_intensities.gain** (associated esd).

[array_intensities]

***_array_intensities.linearity** (*code*)

The intensity linearity scaling method used to convert from the raw intensity to the stored element value. 'linear' is linear. 'offset' means that the value defined by **_array_intensities.offset** should be added to each element value. 'scaling' means that the value defined by **_array_intensities.scaling** should be multiplied with each element value. 'scaling_offset' is the combination of the two previous cases, with the scale factor applied before the offset value. 'sqrt_scaled' means that the square root of raw intensities multiplied by **_array_intensities.scaling** is calculated and stored, perhaps rounded to the nearest integer. Thus, linearization involves dividing the stored values by **_array_intensities.scaling** and squaring the result. 'logarithmic_scaled' means that the logarithm base 10 of raw intensities multiplied by **_array_intensities.scaling** is calculated and stored, perhaps rounded to the nearest integer. Thus, linearization involves dividing the stored values by **_array_intensities.scaling** and calculating 10 to the power of this number. 'raw' means that the data are a set of raw values straight from the detector.

The data value must be one of the following:

linear	Linear.
offset	The value defined by **_array_intensities.offset** should be added to each element value.
scaling	The value defined by **_array_intensities.scaling** should be multiplied with each element value.
scaling_offset	The combination of scaling and offset with the scale factor applied before the offset value.
sqrt_scaled	The square root of raw intensities multiplied by **_array_intensities.scaling** is calculated and stored, perhaps rounded to the nearest integer. Thus, linearization involves dividing the stored values by **_array_intensities.scaling** and squaring the result.
logarithmic_scaled	The logarithm base 10 of raw intensities multiplied by **_array_intensities.scaling** is calculated and stored, perhaps rounded to the nearest integer. Thus, linearization involves dividing the stored values by **_array_intensities.scaling** and calculating 10 to the power of this number.
raw	The array consists of raw values to which no corrections have been applied. While the handling of the data is similar to that given for 'linear' data with no offset, the meaning of the data differs in that the number of incident photons is not necessarily linearly related to the number of counts reported. This value is intended for use either in calibration experiments or to allow for handling more complex data-fitting algorithms than are allowed for by this data item.

[array_intensities]

_array_intensities.offset (*float*)

Offset value to add to array element values in the manner described by the item **_array_intensities.linearity**.

[array_intensities]

_array_intensities.overload (*float*)

The saturation intensity level for this data array.

[array_intensities]

_array_intensities.scaling (*float*)

Multiplicative scaling value to be applied to array data in the manner described by item **_array_intensities.linearity**.

[array_intensities]

_array_intensities.undefined_value (*float*)

A value to be substituted for undefined values in the data array.

[array_intensities]

ARRAY_STRUCTURE

Data items in the ARRAY_STRUCTURE category record the organization and encoding of array data in the ARRAY_DATA category.

Category group(s): **inclusive_group**
array_data_group

Category key(s): **_array_structure.id**

Example 1.

```
loop_
_array_structure.id
_array_structure.encoding_type
_array_structure.compression_type
_array_structure.byte_order
 image_1   "unsigned 16-bit integer"  none  little_endian
```

***_array_structure.byte_order** (*code*)

The order of bytes for integer values which require more than 1 byte. (IBM PCs and compatibles, and Dec VAXs use low-byte-first ordered integers, whereas Hewlett Packard 700 series, Sun-4 and Silicon Graphics use high-byte-first ordered integers. Dec Alphas can produce/use either depending on a compiler switch.)

The data value must be one of the following:

big_endian	The first byte in the byte stream of the bytes which make up an integer value is the most significant byte of an integer.
little_endian	The last byte in the byte stream of the bytes which make up an integer value is the most significant byte of an integer.

[array_structure]

_array_structure.compression_type (*code*)

Type of data-compression method used to compress the array data.

The data value must be one of the following:

none	Data are stored in normal format as defined by **_array_structure.encoding_type** and **_array_structure.byte_order**.
packed	Using the 'packed' compression scheme, a CCP4-style packing (*International Tables for Crystallography* Volume G, Section 5.6.3.2)
canonical	Using the 'canonical' compression scheme (*International Tables for Crystallography* Volume G, Section 5.6.3.1)

Where no value is given, the assumed value is 'none'. **[array_structure]**

***_array_structure.encoding_type** (*uline*)

Data encoding of a single element of array data. In several cases, the IEEE format is referenced. See IEEE Standard 754-1985 (IEEE, 1985).

Reference: IEEE (1985). *IEEE Standard for Binary Floating-Point Arithmetic*. ANSI/IEEE Std 754-1985. New York: Institute of Electrical and Electronics Engineers.

The data value must be one of the following:

'unsigned 8-bit integer'
'signed 8-bit integer'
'unsigned 16-bit integer'
'signed 16-bit integer'
'unsigned 32-bit integer'
'signed 32-bit integer'
'signed 32-bit real IEEE'
'signed 64-bit real IEEE'
'signed 32-bit complex IEEE' **[array_structure]**

***_array_structure.id** (*code*)

The value of **_array_structure.id** must uniquely identify each item of array data.

The following item(s) have an equivalent role in their respective categories:

_array_data.array_id,
_array_structure_list.array_id,
_array_intensities.array_id,
_diffrn_data_frame.array_id. **[array_structure]**

ARRAY_STRUCTURE_LIST

Data items in the ARRAY_STRUCTURE_LIST category record the size and organization of each array dimension. The relationship to physical axes may be given.

Category group(s): **inclusive_group**
array_data_group
Category key(s): **_array_structure_list.array_id**
_array_structure_list.index

Example 1 – an image array of 1300 × 1200 elements.
The raster order of the image is left to right (increasing) in the first dimension and bottom to top (decreasing) in the second dimension.

```
loop_
 _array_structure_list.array_id
 _array_structure_list.index
 _array_structure_list.dimension
 _array_structure_list.precedence
 _array_structure_list.direction
 _array_structure_list.axis_set_id
  image_1   1    1300    1     increasing  ELEMENT_X
  image_1   2    1200    2     decreasing  ELEMENY_Y
```

***_array_structure_list.array_id** (*code*)
This item is a pointer to **_array_structure.id** in the ARRAY_STRUCTURE category.

[array_structure_list]

***_array_structure_list.axis_set_id** (*code*)
This is a descriptor for the physical axis or set of axes corresponding to an array index. This data item is related to the axes of the detector itself given in DIFFRN_DETECTOR_AXIS, but usually differs in that the axes in this category are the axes of the coordinate system of reported data points, while the axes in DIFFRN_DETECTOR_AXIS are the physical axes of the detector describing the 'poise' of the detector as an overall physical object. If there is only one axis in the set, the identifier of that axis should be used as the identifier of the set.

The following item(s) have an equivalent role in their respective categories:
_array_structure_list_axis.axis_set_id. **[array_structure_list]**

***_array_structure_list.dimension** (*int*)
The number of elements stored in the array structure in this dimension.
The permitted range is $[1, \infty)$. **[array_structure_list]**

***_array_structure_list.direction** (*int*)
Identifies the direction in which this array index changes.
The data value must be one of the following:

increasing	Indicates the index changes from 1 to the maximum dimension
decreasing	Indicates the index changes from the maximum dimension to 1

[array_structure_list]

***_array_structure_list.index** (*int*)
Identifies the one-based index of the row or column in the array structure.

The following item(s) have an equivalent role in their respective categories:
_array_element_size.index.
The permitted range is $[1, \infty)$. **[array_structure_list]**

***_array_structure_list.precedence** (*int*)
Identifies the rank order in which this array index changes with respect to other array indices. The precedence of 1 indicates the index which changes fastest.
The permitted range is $[1, \infty)$. **[array_structure_list]**

ARRAY_STRUCTURE_LIST_AXIS

Data items in the ARRAY_STRUCTURE_LIST_AXIS category describe the physical settings of sets of axes for the centres of pixels that correspond to data points described in the ARRAY_STRUCTURE_LIST category. In the simplest cases, the physical increments of a single axis correspond to the increments of a single array index. More complex organizations, *e.g.* spiral scans, may require coupled motions along multiple axes. Note that a spiral scan uses two coupled axes: one for the angular direction and one for the radial direction. This differs from a cylindrical scan for which the two axes are not coupled into one set.

Category group(s): **inclusive_group**
array_data_group
Category key(s): **_array_structure_list_axis.axis_set_id**
_array_structure_list_axis.axis_id

_array_structure_list_axis.angle (*float*)
The setting of the specified axis in degrees for the first data point of the array index with the corresponding value of **_array_structure_list.axis_set_id**. If the index is specified as 'increasing', this will be the centre of the pixel with index value 1. If the index is specified as 'decreasing', this will be the centre of the pixel with maximum index value.
Where no value is given, the assumed value is '0.0'.

[array_structure_list_axis]

_array_structure_list_axis.angle_increment (*float*)
The pixel-centre-to-pixel-centre increment in the angular setting of the specified axis in degrees. This is not meaningful in the case of 'constant velocity' spiral scans and should not be specified for this case. See **_array_structure_list_axis.angular_pitch**.
Where no value is given, the assumed value is '0.0'.

[array_structure_list_axis]

_array_structure_list_axis.angular_pitch (*float*)
The pixel-centre-to-pixel-centre distance for a one-step change in the setting of the specified axis in millimetres. This is meaningful only for 'constant velocity' spiral scans or for uncoupled angular scans at a constant radius (cylindrical scans) and should not be specified for cases in which the angle between pixels (rather than the distance between pixels) is uniform. See **_array_structure_list_axis.angle_increment**.
Where no value is given, the assumed value is '0.0'.

[array_structure_list_axis]

***_array_structure_list_axis.axis_id** (*code*)
The value of this data item is the identifier of one of the axes in the set of axes for which settings are being specified. Multiple axes may be specified for the same value of **_array_structure_list_axis.axis_set_id**. This item is a pointer to **_axis.id** in the AXIS category.

[array_structure_list_axis]

(*)**_array_structure_list_axis.axis_set_id** (*code*)
The value of this data item is the identifier of the set of axes for which axis settings are being specified. Multiple axes may be specified for the same value of **_array_structure_list_axis.axis_set_id**. This item is a pointer to **_array_structure_list.axis_set_id** in the ARRAY_STRUCTURE_LIST category. If this item is not specified, it defaults to the corresponding axis identifier.

[array_structure_list_axis]

_array_structure_list_axis.displacement (*float*)
The setting of the specified axis in millimetres for the first data point of the array index with the corresponding value of **_array_structure_list.axis_set_id**. If the index is specified as 'increasing', this will be the centre of the pixel with index value 1. If the index is specified as 'decreasing', this will be the centre of the pixel with maximum index value.
Where no value is given, the assumed value is '0.0'.

[array_structure_list_axis]

_array_structure_list_axis.displacement_increment (*float*)

The pixel-centre-to-pixel-centre increment for the displacement setting of the specified axis in millimetres.

Where no value is given, the assumed value is '0.0'.

[array_structure_list_axis]

_array_structure_list_axis.radial_pitch (*float*)

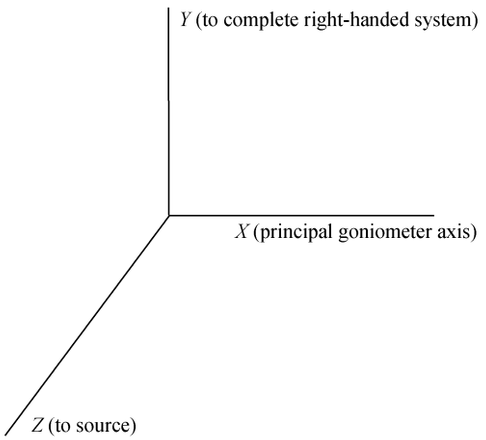
The radial distance from one 'cylinder' of pixels to the next in millimetres. If the scan is a 'constant velocity' scan with differing angular displacements between pixels, the value of this item may differ significantly from the value of **_array_structure_list_axis.displacement_increment**.

Where no value is given, the assumed value is '0.0'.

[array_structure_list_axis]

AXIS

Data items in the AXIS category record the information required to describe the various goniometer, detector, source and other axes needed to specify a data collection. The location of each axis is specified by two vectors: the axis itself, given as a unit vector, and an offset to the base of the unit vector. These vectors are referenced to a right-handed laboratory coordinate system with its origin in the sample or specimen:



Axis 1 (X): The X axis is aligned to the mechanical axis pointing from the sample or specimen along the principal axis of the goniometer. Axis 2 (Y): The Y axis completes an orthogonal right-handed system defined by the X axis and the Z axis (see below). Axis 3 (Z): The Z axis is derived from the source axis which goes from the sample to the source. The Z axis is the component of the source axis in the direction of the source orthogonal to the X axis in the plane defined by the X axis and the source axis. These axes are based on the goniometer, not on the orientation of the detector, gravity *etc.* The vectors necessary to specify all other axes are given by sets of three components in the order (X, Y, Z). If the axis involved is a rotation axis, it is right-handed, *i.e.* as one views the object to be rotated from the origin (the tail) of the unit vector, the rotation is clockwise. If a translation axis is specified, the direction of the unit vector specifies the sense of positive translation. *Note:* This choice of coordinate system is similar to but significantly different from the choice in *MOSFLM* (Leslie & Powell, 2004). In *MOSFLM*, X is along the X-ray beam (the CBF/imgCIF Z axis) and Z is along the rotation axis.

All rotations are given in degrees and all translations are given in millimetres. Axes may be dependent on one another. The X axis is the only goniometer axis the direction of which is strictly connected to the hardware. All other axes are specified by the positions they would assume when the axes upon which they depend are at their zero points.

When specifying detector axes, the axis is given to the beam centre. The location of the beam centre on the detector should be given in the DIFFRN_DETECTOR category in distortion-corrected millimetres from the (0, 0) corner of the detector. It should be noted that many different origins arise in the definition of an experiment. In particular, as noted above, it is necessary to specify the location of the beam centre on the detector in terms of the origin of the detector, which is, of course, not coincident with the centre of the sample.

Reference: Leslie, A. G. W. & Powell, H. (2004). *MOSFLM* v6.11. MRC Laboratory of Molecular Biology, Hills Road, Cambridge, England. http://www.CCP4.ac.uk/dist/x-windows/Mosflm/.

Category group(s): **inclusive_group**
axis_group
diffrn_group

Category key(s): **_axis.id**
_axis.equipment

Example 1.

This example shows the axis specification of the axes of a kappa-geometry goniometer [see Stout, G. H. & Jensen, L. H. (1989). *X-ray structure determination. A practical guide*, 2nd ed. p. 134. New York: Wiley Interscience]. There are three axes specified, and no offsets. The outermost axis, ω, is pointed along the X axis. The next innermost axis, κ, is at a 50° angle to the X axis, pointed away from the source. The innermost axis, φ, aligns with the X axis when ω and φ are at their zero points. If T_ω, T_κ and T_φ are the transformation matrices derived from the axis settings, the complete transformation would be $x' = T_\omega T_\kappa T_\varphi x$.

```
loop_
  _axis.id
  _axis.type
  _axis.equipment
  _axis.depends_on
  _axis.vector[1] _axis.vector[2] _axis.vector[3]
omega rotation goniometer     .    1        0        0
kappa rotation goniometer omega    -.64279  0       -.76604
phi   rotation goniometer kappa    1        0        0
```

Example 2.

This example show the axis specification of the axes of a detector, source and gravity. The order has been changed as a reminder that the ordering of presentation of tokens is not significant. The centre of rotation of the detector has been taken to be 68 mm in the direction away from the source.

```
loop_
  _axis.id
  _axis.type
  _axis.equipment
  _axis.depends_on
  _axis.vector[1] _axis.vector[2] _axis.vector[3]
  _axis.offset[1] _axis.offset[2] _axis.offset[3]
source       .        source     .        0   0   1  . . .
gravity      .        gravity    .        0  -1   0  . . .
tranz     translation detector rotz       0   0   1  0 0 -68
twotheta  rotation    detector   .        1   0   0  . . .
roty      rotation    detector twotheta   0   1   0  0 0 -68
rotz      rotation    detector roty       0   0   1  0 0 -68
```

_axis.depends_on

The value of **_axis.depends_on** specifies the next outermost axis upon which this axis depends. This item is a pointer to **_axis.id** in the same category.

[axis]

_axis.equipment (*ucode*)

The value of **_axis.equipment** specifies the type of equipment using the axis: 'goniometer', 'detector', 'gravity', 'source' or 'general'.

The data value must be one of the following:

goniometer	equipment used to orient or position samples
detector	equipment used to detect reflections
general	equipment used for general purposes
gravity	axis specifying the downward direction
source	axis specifying the direction sample to source

Where no value is given, the assumed value is 'general'. **[axis]**

*`_axis.id` (*code*)

The value of `_axis.id` must uniquely identify each axis relevant to the experiment. Note that multiple pieces of equipment may share the same axis (*e.g.* a 2θ arm), so the category key for AXIS also includes the equipment.

The following item(s) have an equivalent role in their respective categories:

`_axis.depends_on`,
`_array_structure_list_axis.axis_id`,
`_diffrn_detector_axis.axis_id`,
`_diffrn_measurement_axis.axis_id`,
`_diffrn_scan_axis.axis_id`,
`_diffrn_scan_frame_axis.axis_id`. **[axis]**

`_axis.offset[1]` (*float*)

The [1] element of the three-element vector used to specify the offset to the base of a rotation or translation axis. The vector is specified in millimetres.

Where no value is given, the assumed value is '`0.0`'. **[axis]**

`_axis.offset[2]` (*float*)

The [2] element of the three-element vector used to specify the offset to the base of a rotation or translation axis. The vector is specified in millimetres.

Where no value is given, the assumed value is '`0.0`'. **[axis]**

`_axis.offset[3]` (*float*)

The [3] element of the three-element vector used to specify the offset to the base of a rotation or translation axis. The vector is specified in millimetres.

Where no value is given, the assumed value is '`0.0`'. **[axis]**

`_axis.type` (*ucode*)

The value of `_axis.type` specifies the type of axis: 'rotation' or 'translation' (or 'general' when the type is not relevant, as for gravity).

The data value must be one of the following:

`rotation`	right-handed axis of rotation
`translation`	translation in the direction of the axis
`general`	axis for which the type is not relevant

Where no value is given, the assumed value is '`general`'. **[axis]**

`_axis.vector[1]` (*float*)

The [1] element of the three-element vector used to specify the direction of a rotation or translation axis. The vector should be normalized to be a unit vector and is dimensionless.

Where no value is given, the assumed value is '`0.0`'. **[axis]**

`_axis.vector[2]` (*float*)

The [2] element of the three-element vector used to specify the direction of a rotation or translation axis. The vector should be normalized to be a unit vector and is dimensionless.

Where no value is given, the assumed value is '`0.0`'. **[axis]**

`_axis.vector[3]` (*float*)

The [3] element of the three-element vector used to specify the direction of a rotation or translation axis. The vector should be normalized to be a unit vector and is dimensionless.

Where no value is given, the assumed value is '`0.0`'. **[axis]**

DIFFRN_DATA_FRAME

Data items in the DIFFRN_DATA_FRAME category record the details about each frame of data. The items in this category were previously in a DIFFRN_FRAME_DATA category, which is now deprecated. The items from the old category are provided as aliases but should not be used for new work.

Category group(s): `inclusive_group`
`array_data_group`
Category key(s): `_diffrn_data_frame.id`
`_diffrn_data_frame.detector_element_id`

Example 1 – a frame containing data from four frame elements.

Each frame element has a common array configuration 'array_1' described in ARRAY_STRUCTURE and related categories. The data for each detector element are stored in four groups of binary data in the ARRAY_DATA category, linked by the array_id and binary_id.

```
loop_
  _diffrn_data_frame.id
  _diffrn_data_frame.detector_element_id
  _diffrn_data_frame.array_id
  _diffrn_data_frame.binary_id
frame_1   d1_ccd_1  array_1  1
frame_1   d1_ccd_2  array_1  2
frame_1   d1_ccd_3  array_1  3
frame_1   d1_ccd_4  array_1  4
```

*`_diffrn_data_frame.array_id` (*code*)

`_diffrn_frame_data.array_id` (*cif_img.dic 1.0*)

This item is a pointer to `_array_structure.id` in the ARRAY_STRUCTURE category. **[diffrn_data_frame]**

(*)`_diffrn_data_frame.binary_id` (*int*)

`_diffrn_frame_data.binary_id` (*cif_img.dic 1.0*)

This item is a pointer to `_array_data.binary_id` in the ARRAY_DATA category. **[diffrn_data_frame]**

*`_diffrn_data_frame.detector_element_id` (*code*)

`_diffrn_frame_data.detector_element_id` (*cif_img.dic 1.0*)

This item is a pointer to `_diffrn_detector_element.id` in the DIFFRN_DETECTOR_ELEMENT category. **[diffrn_data_frame]**

*`_diffrn_data_frame.id` (*code*)

`_diffrn_frame_data.id` (*cif_img.dic 1.0*)

The value of `_diffrn_data_frame.id` must uniquely identify each complete frame of data.

The following item(s) have an equivalent role in their respective categories:

`_diffrn_refln.frame_id`,
`_diffrn_scan.frame_id_start`,
`_diffrn_scan.frame_id_end`,
`_diffrn_scan_frame.frame_id`,
`_diffrn_scan_frame_axis.frame_id`. **[diffrn_data_frame]**

DIFFRN_DETECTOR

Data items in the DIFFRN_DETECTOR category describe the detector used to measure the scattered radiation, including any analyser and post-sample collimation.

Category group(s): `inclusive_group`
`diffrn_group`
Category key(s): `_diffrn_detector.diffrn_id`
`_diffrn_detector.id`

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_detector.diffrn_id      'd1'
_diffrn_detector.detector       'multiwire'
_diffrn_detector.type           'Siemens'
```

`_diffrn_detector.details` (*text*)

`_diffrn_detector_details` (*cif_core.dic 2.0.1*)

A description of special aspects of the radiation detector.

Example: '`slow mode`'. **[diffrn_detector]**

_diffrn_detector.detector (*text*)
_diffrn_radiation_detector (*cifdic.c91 1.0*)
_diffrn_detector (*cif_core.dic 2.0*)
The general class of the radiation detector.
Examples: 'photographic film', 'scintillation counter', 'CCD plate', 'BF~3~ counter'. **[diffrn_detector]**

***_diffrn_detector.diffrn_id** (*code*)
This data item is a pointer to **_diffrn.id** in the DIFFRN category. The value of **_diffrn.id** uniquely defines a set of diffraction data.

_diffrn_detector.dtime (*float*)
_diffrn_radiation_detector_dtime (*cifdic.c91 1.0*)
_diffrn_detector_dtime (*cif_core.dic 2.0*)
The deadtime in microseconds of the detector(s) used to measure the diffraction intensities.
The permitted range is $[0.0, \infty)$. **[diffrn_detector]**

(*)**_diffrn_detector.id** (*code*)
The value of **_diffrn_detector.id** must uniquely identify each detector used to collect each diffraction data set. If the value of **_diffrn_detector.id** is not given, it is implicitly equal to the value of **_diffrn_detector.diffrn_id**.
The following item(s) have an equivalent role in their respective categories:
_diffrn_detector_axis.detector_id. **[diffrn_detector]**

_diffrn_detector.number_of_axes (*int*)
The value of **_diffrn_detector.number_of_axes** gives the number of axes of the positioner for the detector identified by **_diffrn_detector.id**. The word 'positioner' is a general term used in instrumentation design for devices that are used to change the positions of portions of apparatus by linear translation, rotation or combinations of such motions. Axes which are used to provide a coordinate system for the face of an area detetctor should not be counted for this data item. The description of each axis should be provided by entries in DIFFRN_DETECTOR_AXIS.
The permitted range is $[1, \infty)$. **[diffrn_detector]**

_diffrn_detector.type (*text*)
_diffrn_detector_type (*cif_core.dic 2.0.1*)
The make, model or name of the detector device used.
[diffrn_detector]

DIFFRN_DETECTOR_AXIS

Data items in the DIFFRN_DETECTOR_AXIS category associate axes with detectors.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_detector_axis.detector_id**
_diffrn_detector_axis.axis_id

***_diffrn_detector_axis.axis_id** (*code*)
This data item is a pointer to **_axis.id** in the AXIS category.
[diffrn_detector_axis]

***_diffrn_detector_axis.detector_id** (*code*)
_diffrn_detector_axis.id (*cif_img.dic 1.0*)
This data item is a pointer to **_diffrn_detector.id** in the DIFFRN_DETECTOR category. This item was previously named **_diffrn_detector_axis.id**, which is now a deprecated name. The old name is provided as an alias but should not be used for new work.
[diffrn_detector_axis]

***_diffrn_detector_axis.id** (*code*)
This data item is a pointer to **_diffrn_detector.id** in the DIFFRN_DETECTOR category. *Deprecated: do not use.*
[diffrn_detector_axis]

DIFFRN_DETECTOR_ELEMENT

Data items in the DIFFRN_DETECTOR_ELEMENT category record the details about spatial layout and other characteristics of each element of a detector which may have multiple elements. In most cases, giving more detailed information in ARRAY_STRUCTURE_LIST and ARRAY_STRUCTURE_LIST_AXIS is preferable to simply providing the centre of the detector element.
Category group(s): **inclusive_group**
array_data_group
Category key(s): **_diffrn_detector_element.id**
_diffrn_detector_element.detector_id

Example 1
Detector d1 is composed of four CCD detector elements, each 200 by 200 mm, arranged in a square, in the pattern

```
1      2
   *
3      4
```

Note that the beam centre is slightly displaced from each of the detector elements, just beyond the lower right corner of 1, the lower left corner of 2, the upper right corner of 3 and the upper left corner of 4.

```
loop_
  _diffrn_detector_element.detector_id
  _diffrn_detector_element.id
  _diffrn_detector_element.center[1]
  _diffrn_detector_element.center[2]
    d1     d1_ccd_1  201.5 -1.5
    d1     d1_ccd_2  -1.8  -1.5
    d1     d1_ccd_3  201.6 201.4
    d1     d1_ccd_4  -1.7  201.5
```

_diffrn_detector_element.center[1] (*float*)
The value of **_diffrn_detector_element.center[1]** is the *X* component of the distortion-corrected beam centre in millimetres from the (0, 0) (lower-left) corner of the detector element viewed from the sample side. The *X* and *Y* axes are the laboratory coordinate system coordinates defined in the AXIS category measured when all positioning axes for the detector are at their zero settings. If the resulting *X* or *Y* axis is then orthogonal to the detector, the *Z* axis is used instead of the orthogonal axis.
Where no value is given, the assumed value is '0.0'. **[diffrn_detector_element]**

_diffrn_detector_element.center[2] (*float*)
The value of **_diffrn_detector_element.center[2]** is the *Y* component of the distortion-corrected beam centre in millimetres from the (0, 0) (lower-left) corner of the detector element viewed from the sample side. The *X* and *Y* axes are the laboratory coordinate system coordinates defined in the AXIS category measured when all positioning axes for the detector are at their zero settings. If the resulting *X* or *Y* axis is then orthogonal to the detector, the *Z* axis is used instead of the orthogonal axis.
Where no value is given, the assumed value is '0.0'. **[diffrn_detector_element]**

***_diffrn_detector_element.detector_id** (*code*)
This item is a pointer to **_diffrn_detector.id** in the DIFFRN_DETECTOR category.
[diffrn_detector_element]

***_diffrn_detector_element.id** (*code*)
The value of **_diffrn_detector_element.id** must uniquely identify each element of a detector.
[diffrn_detector_element]

DIFFRN_FRAME_DATA

Data items in the DIFFRN_FRAME_DATA category record the details about each frame of data. The items in this category are now in the DIFFRN_DATA_FRAME category. The items in the DIFFRN_FRAME_DATA category are now deprecated. The items from this category are provided as aliases in version 1.0 of the dictionary but should not be used for new work. The items from the old category are provided in this dictionary for completeness but should not be used or cited. To avoid confusion, the example has been removed and the redundant parent–child links to other categories have been removed.

Category group(s): **inclusive_group**
array_data_group
Category key(s): **_diffrn_frame_data.id**
_diffrn_frame_data.detector_element_id

THE DIFFRN_FRAME_DATA category is deprecated and should not be used.

```
# EXAMPLE REMOVED #
```

*_diffrn_frame_data.array_id (*code*)

This item is a pointer to **_array_structure.id** in the ARRAY_STRUCTURE category. *Deprecated: do not use.*

[diffrn_frame_data]

(*)_diffrn_frame_data.binary_id (*int*)

This item is a pointer to **_array_data.binary_id** in the ARRAY_STRUCTURE category. *Deprecated: do not use.*

[diffrn_frame_data]

*_diffrn_frame_data.detector_element_id (*code*)

This item is a pointer to **_diffrn_detector_element.id** in the DIFFRN_DETECTOR_ELEMENT category. *Deprecated: do not use.*

[diffrn_frame_data]

*_diffrn_frame_data.id (*code*)

The value of **_diffrn_frame_data.id** must uniquely identify each complete frame of data. *Deprecated: do not use.*

[diffrn_frame_data]

DIFFRN_MEASUREMENT

Data items in the DIFFRN_MEASUREMENT category record details about the device used to orient and/or position the crystal during data measurement and the manner in which the diffraction data were measured.

Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_measurement.device**
_diffrn_measurement.diffrn_id
_diffrn_measurement.id

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_measurement.diffrn_id          'd1'
_diffrn_measurement.device             '3-circle camera'
_diffrn_measurement.device_type        'Supper model x'
_diffrn_measurement.device_details     'none'
_diffrn_measurement.method             'omega scan'
_diffrn_measurement.details
; 440 frames, 0.20 degrees, 150 sec, detector distance 12 cm,
  detector angle 22.5 degrees
;
```

*Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C**47**, 2276–2277].*

```
_diffrn_measurement.diffrn_id      's1'
_diffrn_measurement.device_type
     'Philips PW1100/20 diffractometer'
_diffrn_measurement.method         'theta/2theta (\q/2\q)'
```

_diffrn_measurement.details (*text*)

_diffrn_measurement_details (*cif_core.dic 2.0.1*)

A description of special aspects of the intensity measurement.

Example:

```
; 440 frames, 0.20 degrees, 150 sec, detector
  distance 12 cm, detector angle 22.5 degrees
;
```

[diffrn_measurement]

(*)_diffrn_measurement.device (*text*)

_diffrn_measurement_device (*cif_core.dic 2.0.1*)

The general class of goniometer or device used to support and orient the specimen. If the value of **_diffrn_measurement.device** is not given, it is implicitly equal to the value of **_diffrn_measurement.diffrn_id**. Either **_diffrn_measurement.device** or **_diffrn_measurement.id** may be used to link to other categories. If the experimental setup admits multiple devices, then **_diffrn_measurement.id** is used to provide a unique link.

The following item(s) have an equivalent role in their respective categories:

_diffrn_measurement_axis.measurement_device.

Examples: '3-circle camera', '4-circle camera', 'kappa-geometry camera', 'oscillation camera', 'precession camera'.

[diffrn_measurement]

_diffrn_measurement.device_details (*text*)

_diffrn_measurement_device_details (*cif_core.dic 2.0.1*)

A description of special aspects of the device used to measure the diffraction intensities.

Example:

```
; commercial goniometer modified locally to allow for 90\% \t
arc
;
```

[diffrn_measurement]

_diffrn_measurement.device_type (*text*)

_diffrn_measurement_device_type (*cif_core.dic 2.0.1*)

The make, model or name of the measurement device (goniometer) used.

Examples: 'Supper model q', 'Huber model r', 'Enraf-Nonius model s', 'home-made'. **[diffrn_measurement]**

*_diffrn_measurement.diffrn_id

This data item is a pointer to **_diffrn.id** in the DIFFRN category.

(*)_diffrn_measurement.id (*code*)

The value of **_diffrn_measurement.id** must uniquely identify the set of mechanical characteristics of the device used to orient and/or position the sample used during the collection of each diffraction data set. If the value of **_diffrn_measurement.id** is not given, it is implicitly equal to the value of **_diffrn_measurement.diffrn_id**. Either **_diffrn_measurement.device** or **_diffrn_measurement.id** may be used to link to other categories. If the experimental setup admits multiple devices, then **_diffrn_measurement.id** is used to provide a unique link.

The following item(s) have an equivalent role in their respective categories:

_diffrn_measurement_axis.measurement_id. **[diffrn_measurement]**

_diffrn_measurement.method (*text*)

_diffrn_measurement_method (*cif_core.dic 2.0.1*)

Method used to measure intensities.

Example: 'profile data from theta/2theta (\q/2\q) scans'.

[diffrn_measurement]

_diffrn_measurement.number_of_axes (*int*)

The value of **_diffrn_measurement.number_of_axes** gives the number of axes of the positioner for the goniometer or other sample orientation or positioning device identified by **_diffrn_measurement.id**. The description of the axes should be provided by entries in DIFFRN_MEASUREMENT_AXIS.

The permitted range is $[1, \infty)$. **[diffrn_measurement]**

_diffrn_measurement.specimen_support (*text*)
_diffrn_measurement_specimen_support (*cif_core.dic 2.0.1*)
The physical device used to support the crystal during data collection.
Examples: 'glass capillary', 'quartz capillary', 'fiber', 'metal loop'.
[diffrn_measurement]

DIFFRN_MEASUREMENT_AXIS

Data items in the DIFFRN_MEASUREMENT_AXIS category associate axes with goniometers.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_measurement_axis.measurement_device**
_diffrn_measurement_axis.measurement_id
_diffrn_measurement_axis.axis_id

***_diffrn_measurement_axis.axis_id** (*code*)
This data item is a pointer to **_axis.id** in the AXIS category.
[diffrn_measurement_axis]

***_diffrn_measurement_axis.id** (*code*)
This data item is a pointer to **_diffrn_measurement.id** in the DIFFRN_MEASUREMENT category. *Deprecated: do not use.*
[diffrn_measurement_axis]

(*)**_diffrn_measurement_axis.measurement_device** (*text*)
This data item is a pointer to **_diffrn_measurement.device** in the DIFFRN_MEASUREMENT category.
[diffrn_measurement_axis]

(*)**_diffrn_measurement_axis.measurement_id** (*code*)
_diffrn_measurement_axis.id (*cif_img.dic 1.0*)
This data item is a pointer to **_diffrn_measurement.id** in the DIFFRN_MEASUREMENT category. This item was previously named **_diffrn_measurement_axis.id**, which is now a deprecated name. The old name is provided as an alias but should not be used for new work.
[diffrn_measurement_axis]

DIFFRN_RADIATION

Data items in the DIFFRN_RADIATION category describe the radiation used for measuring diffraction intensities, its collimation and monochromatization before the sample. Post-sample treatment of the beam is described by data items in the DIFFRN_DETECTOR category.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_radiation.diffrn_id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
_diffrn_radiation.diffrn_id         'set1'
_diffrn_radiation.collimation       '0.3 mm double pinhole'
_diffrn_radiation.monochromator     'graphite'
_diffrn_radiation.type              'Cu K\a'
_diffrn_radiation.wavelength_id      1
```

Example 2 – based on data set TOZ of Willis, Beckwith & Tozer [Acta Cryst. (1991), C47, 2276–2277].

```
_diffrn_radiation.wavelength_id   1
_diffrn_radiation.type            'Cu K\a'
_diffrn_radiation.monochromator   'graphite'
```

_diffrn_radiation.collimation (*text*)
_diffrn_radiation_collimation (*cif_core.dic 2.0.1*)
The collimation or focusing applied to the radiation.
Examples: '0.3 mm double-pinhole', '0.5 mm', 'focusing mirrors'.
[diffrn_radiation]

***_diffrn_radiation.diffrn_id** (*code*)
This data item is a pointer to **_diffrn.id** in the DIFFRN category.

_diffrn_radiation.div_x_source (*float*)
Beam crossfire in degrees parallel to the laboratory X axis (see AXIS category). This is a characteristic of the X-ray beam as it illuminates the sample (or specimen) after all monochromation and collimation. This is the standard uncertainty (e.s.d.) of the directions of photons in the XZ plane around the mean source beam direction. Note that for some synchrotrons this value is specified in milliradians, in which case a conversion is needed. To convert a value in milliradians to a value in degrees, multiply by 0.180 and divide by π.
[diffrn_radiation]

_diffrn_radiation.div_x_y_source (*float*)
Beam crossfire correlation in degrees squared between the crossfire laboratory X axis component and the crossfire laboratory Y axis component (see AXIS category). This is a characteristic of the X-ray beam as it illuminates the sample (or specimen) after all monochromation and collimation. This is the mean of the products of the deviations of the direction of each photon in the XZ plane times the deviations of the direction of the same photon in the YZ plane around the mean source beam direction. This will be zero for uncorrelated crossfire. Note that for some synchrotrons this value is specified in milliradians squared, in which case a conversion is needed. To convert a value in milliradians squared to a value in degrees squared, multiply by 0.180^2 and divide by π^2.
Where no value is given, the assumed value is '0.0'. **[diffrn_radiation]**

_diffrn_radiation.div_y_source (*float*)
Beam crossfire in degrees parallel to the laboratory Y axis (see AXIS category). This is a characteristic of the X-ray beam as it illuminates the sample (or specimen) after all monochromation and collimation. This is the standard uncertainty (e.s.d.) of the directions of photons in the YZ plane around the mean source beam direction. Note that for some synchrotrons this value is specified in milliradians, in which case a conversion is needed. To convert a value in milliradians to a value in degrees, multiply by 0.180 and divide by π.
Where no value is given, the assumed value is '0.0'. **[diffrn_radiation]**

_diffrn_radiation.filter_edge (*float*)
_diffrn_radiation_filter_edge (*cif_core.dic 2.0.1*)
Absorption edge in ångströms of the radiation filter used.
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.inhomogeneity (*float*)
_diffrn_radiation_inhomogeneity (*cif_core.dic 2.0.1*)
Half-width in millimetres of the incident beam in the direction perpendicular to the diffraction plane.
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.monochromator (*text*)
_diffrn_radiation_monochromator (*cif_core.dic 2.0.1*)
The method used to obtain monochromatic radiation. If a monochromator crystal is used, the material and the indices of the Bragg reflection are specified.
Examples: 'Zr filter', 'Ge 220', 'none', 'equatorial mounted graphite'.
[diffrn_radiation]

_diffrn_radiation.polarisn_norm (*float*)
_diffrn_radiation_polarisn_norm (*cif_core.dic 2.0.1*)
The angle in degrees, as viewed from the specimen, between the perpendicular component of the polarization and the diffraction plane. See **_diffrn_radiation_polarisn_ratio**.
The permitted range is $[-90.0, 90.0]$. **[diffrn_radiation]**

_diffrn_radiation.polarisn_ratio (*float*)
_diffrn_radiation_polarisn_ratio (*cif_core.dic 2.0.1*)
Polarization ratio of the diffraction beam incident on the crystal. This is the ratio of the perpendicularly polarized to the parallel polarized component of the radiation. The perpendicular component forms an angle of **_diffrn_radiation.polarisn_norm** to the normal to the diffraction plane of the sample (*i.e.* the plane containing the incident and reflected beams).
The permitted range is $[0.0, \infty)$. **[diffrn_radiation]**

_diffrn_radiation.polarizn_source_norm (*float*)
The angle in degrees, as viewed from the specimen, between the normal to the polarization plane and the laboratory Y axis as defined in the AXIS category. Note that this is the angle of polarization of the source photons, either directly from a synchrotron beamline or from a monchromator. This differs from the value of **_diffrn_radiation.polarisn_norm** in that **_diffrn_radiation.polarisn_norm** refers to polarization relative to the diffraction plane rather than to the laboratory axis system. In the case of an unpolarized beam, or a beam with true circular polarization, in which no single plane of polarization can be determined, the plane should be taken as the XZ plane and the angle as 0. See **_diffrn_radiation.polarizn_source_ratio**.
The permitted range is $[-90.0, 90.0]$. Where no value is given, the assumed value is '0.0'.
[diffrn_radiation]

_diffrn_radiation.polarizn_source_ratio (*float*)
$(I_p - I_n)/(I_p + I_n)$, where I_p is the intensity (amplitude squared) of the electric vector in the plane of polarization and I_n is the intensity (amplitude squared) of the electric vector in the plane of the normal to the plane of polarization. In the case of an unpolarized beam, or a beam with true circular polarization, in which no single plane of polarization can be determined, the plane is to be taken as the XZ plane and the normal is parallel to the Y axis. Thus, if there was complete polarization in the plane of polarization, the value of **_diffrn_radiation.polarizn_source_ratio** would be 1, and for an unpolarized beam **_diffrn_radiation.polarizn_source_ratio** would have a value of 0. If the X axis has been chosen to lie in the plane of polarization, this definition will agree with the definition of 'MONOCHROMATOR' in the *Denzo* glossary, and values of near 1 should be expected for a bending-magnet source. However, if the X axis were perpendicular to the polarization plane (not a common choice), then the *Denzo* value would be the negative of **_diffrn_radiation.polarizn_source_ratio**. [See http://www.hkl-xray.com for information on *Denzo*, and Otwinowski & Minor (1997).] This differs both in the choice of ratio and choice of orientation from **_diffrn_radiation.polarisn_ratio**, which, unlike **_diffrn_radiation.polarizn_source_ratio**, is unbounded.

Reference: Otwinowski, Z. & Minor, W. (1997). *Processing of X-ray diffraction data collected in oscillation mode. Methods Enzymol.* **276**, 307–326.
The permitted range is $[-1.0, 1.0]$. **[diffrn_radiation]**

_diffrn_radiation.probe (*line*)
_diffrn_radiation_probe (*cif_core.dic 2.0.1*)
Name of the type of radiation used. It is strongly recommended that this be given so that the probe radiation is clearly specified.
The data value must be one of the following:
- x-ray
- neutron
- electron
- gamma

[diffrn_radiation]

_diffrn_radiation.type (*line*)
_diffrn_radiation_type (*cif_core.dic 2.0.1*)
The nature of the radiation. This is typically a description of the X-ray wavelength in Siegbahn notation.
Examples: 'CuK\a', 'Cu K\a~1~', 'Cu K-L~2,3~', 'white-beam'.
[diffrn_radiation]

***_diffrn_radiation.wavelength_id** (*code*)
This data item is a pointer to **_diffrn_radiation_wavelength.id** in the DIFFRN_RADIATION_WAVELENGTH category.
[diffrn_radiation]

_diffrn_radiation.xray_symbol (*line*)
_diffrn_radiation_xray_symbol (*cif_core.dic 2.0.1*)
The IUPAC symbol for the X-ray wavelength for the probe radiation.
The data value must be one of the following:

K-L~3~	Kα_1 in older Siegbahn notation
K-L~2~	Kα_2 in older Siegbahn notation
K-M~3~	Kβ in older Siegbahn notation
K-L~2,3~	use where K-L$_3$ and K-L$_2$ are not resolved

[diffrn_radiation]

DIFFRN_REFLN

This category redefinition has been added to extend the key of the standard DIFFRN_REFLN category.
Category group(s): **inclusive_group**, **diffrn_group**
Category key(s): **_diffrn_refln.frame_id**

***_diffrn_refln.frame_id** (*code*)
This item is a pointer to **_diffrn_data_frame.id** in the DIFFRN_DATA_FRAME category.
[diffrn_refln]

DIFFRN_SCAN

Data items in the DIFFRN_SCAN category describe the parameters of one or more scans, relating axis positions to frames.
Category group(s): **inclusive_group**, **diffrn_group**
Category key(s): **_diffrn_scan.id**

Example 1 – derived from a suggestion by R. M. Sweet.
The vector of each axis is not given here, because it is provided in the AXIS category. By making **_diffrn_scan_axis.scan_id** and **_diffrn_scan_axis.axis_id** keys of the DIFFRN_SCAN_AXIS category, an arbitrary number of scanning and fixed axes can be specified for a scan. In this example, three rotation axes and one translation axis at nonzero values are specified, with one axis stepping. There is no reason why more axes could not have been specified to step. Range information has been specified, but note that it can be calculated from the number of frames and the increment, so the data item **_diffrn_scan_axis.angle_range** could be dropped. Both the sweep data and the data for a single frame are specified. Note that the information on how the axes are stepped is given twice, once in terms of the overall averages in the value of **_diffrn_scan.integration_time** and the values for DIFFRN_SCAN_AXIS, and precisely for the given frame in the value for **_diffrn_scan_frame.integration_time** and the values for DIFFRN_SCAN_FRAME_AXIS. If dose-related adjustments are made to scan times and nonlinear stepping is done, these values may differ. Therefore, in interpreting the data for a particular frame it is important to use the frame-specific data.

```
_diffrn_scan.id                  1
_diffrn_scan.date_start         '2001-11-18T03:26:42'
_diffrn_scan.date_end           '2001-11-18T03:36:45'
_diffrn_scan.integration_time    3.0
_diffrn_scan.frame_id_start      mad_L2_000
_diffrn_scan.frame_id_end        mad_L2_200
_diffrn_scan.frames              201

loop_
_diffrn_scan_axis.scan_id
_diffrn_scan_axis.axis_id
_diffrn_scan_axis.angle_start
_diffrn_scan_axis.angle_range
_diffrn_scan_axis.angle_increment
_diffrn_scan_axis.displacement_start
_diffrn_scan_axis.displacement_range
_diffrn_scan_axis.displacement_increment
   1 omega 200.0 20.0 0.1 . . .
   1 kappa -40.0  0.0 0.0 . . .
   1 phi   127.5  0.0 0.0 . . .
   1 tranz  . . .   2.3 0.0 0.0
```

```
_diffrn_scan_frame.scan_id            1
_diffrn_scan_frame.date              '2001-11-18T03:27:33'
_diffrn_scan_frame.integration_time   3.0
_diffrn_scan_frame.frame_id           mad_L2_018
_diffrn_scan_frame.frame_number       18

loop_
_diffrn_scan_frame_axis.frame_id
_diffrn_scan_frame_axis.axis_id
_diffrn_scan_frame_axis.angle
_diffrn_scan_frame_axis.angle_increment
_diffrn_scan_frame_axis.displacement
_diffrn_scan_frame_axis.displacement_increment
   mad_L2_018 omega 201.8  0.1 . .
   mad_L2_018 kappa -40.0  0.0 . .
   mad_L2_018 phi   127.5  0.0 . .
   mad_L2_018 tranz  .     .  2.3 0.0
```

Example 2 – a more extensive example (R. M. Sweet, P. J. Ellis & H. J. Bernstein).

A detector is placed 240 mm along the Z axis from the goniometer. This leads to a choice: either the axes of the detector are defined at the origin, and then a Z setting of -240 is entered, or the axes are defined with the necessary Z offset. In this case, the setting is used and the offset is left as zero. This axis is called DETECTOR_Z. The axis for positioning the detector in the Y direction depends on the detector Z axis. This axis is called DETECTOR_Y. The axis for positioning the detector in the X direction depends on the detector Y axis (and therefore on the detector Z axis). This axis is called DETECTOR_X. This detector may be rotated around the Y axis. This rotation axis depends on the three translation axes. It is called DETECTOR_PITCH. A coordinate system is defined on the face of the detector in terms of 2300 0.150 mm pixels in each direction. The ELEMENT_X axis is used to index the first array index of the data array and the ELEMENT_Y axis is used to index the second array index. Because the pixels are 0.150×0.150 mm, the centre of the first pixel is at (0.075, 0.075) in this coordinate system.

```
###CBF: VERSION 1.1

data_image_1

# category DIFFRN
_diffrn.id P6MB
_diffrn.crystal_id P6MB_CRYSTAL7

# category DIFFRN_SOURCE
loop_
_diffrn_source.diffrn_id
_diffrn_source.source
_diffrn_source.type
  P6MB synchrotron 'SSRL beamline 9-1'

# category DIFFRN_RADIATION
loop_
_diffrn_radiation.diffrn_id
_diffrn_radiation.wavelength_id
_diffrn_radiation.monochromator
_diffrn_radiation.polarizn_source_ratio
_diffrn_radiation.polarizn_source_norm
_diffrn_radiation.div_x_source
_diffrn_radiation.div_y_source
_diffrn_radiation.div_x_y_source
  P6MB WAVELENGTH1 'Si 111' 0.8 0.0 0.08 0.01 0.00

# category DIFFRN_RADIATION_WAVELENGTH
loop_
_diffrn_radiation_wavelength.id
_diffrn_radiation_wavelength.wavelength
_diffrn_radiation_wavelength.wt
  WAVELENGTH1 0.98 1.0

# category DIFFRN_DETECTOR
loop_
_diffrn_detector.diffrn_id
_diffrn_detector.id
_diffrn_detector.type
_diffrn_detector.number_of_axes
  P6MB MAR345-SN26 'MAR 345' 4

# category DIFFRN_DETECTOR_AXIS
loop_
_diffrn_detector_axis.detector_id
_diffrn_detector_axis.axis_id
  MAR345-SN26 DETECTOR_X
  MAR345-SN26 DETECTOR_Y
  MAR345-SN26 DETECTOR_Z
  MAR345-SN26 DETECTOR_PITCH

# category DIFFRN_DETECTOR_ELEMENT
loop_
_diffrn_detector_element.id
_diffrn_detector_element.detector_id
  ELEMENT1 MAR345-SN26

# category DIFFRN_DATA_FRAME
loop_
_diffrn_data_frame.id
_diffrn_data_frame.detector_element_id
_diffrn_data_frame.array_id
_diffrn_data_frame.binary_id
  FRAME1 ELEMENT1 ARRAY1 1

# category DIFFRN_MEASUREMENT
loop_
_diffrn_measurement.diffrn_id
_diffrn_measurement.id
_diffrn_measurement.number_of_axes
_diffrn_measurement.method
  P6MB GONIOMETER 3 rotation

# category DIFFRN_MEASUREMENT_AXIS
loop_
_diffrn_measurement_axis.measurement_id
_diffrn_measurement_axis.axis_id
  GONIOMETER GONIOMETER_PHI
  GONIOMETER GONIOMETER_KAPPA
  GONIOMETER GONIOMETER_OMEGA

# category DIFFRN_SCAN
loop_
_diffrn_scan.id
_diffrn_scan.frame_id_start
_diffrn_scan.frame_id_end
_diffrn_scan.frames
  SCAN1 FRAME1 FRAME1 1

# category DIFFRN_SCAN_AXIS
loop_
_diffrn_scan_axis.scan_id
_diffrn_scan_axis.axis_id
_diffrn_scan_axis.angle_start
_diffrn_scan_axis.angle_range
_diffrn_scan_axis.angle_increment
_diffrn_scan_axis.displacement_start
_diffrn_scan_axis.displacement_range
_diffrn_scan_axis.displacement_increment
  SCAN1 GONIOMETER_OMEGA 12.0 1.0 1.0 0.0 0.0 0.0
  SCAN1 GONIOMETER_KAPPA 23.3 0.0 0.0 0.0 0.0 0.0
  SCAN1 GONIOMETER_PHI -165.8 0.0 0.0 0.0 0.0 0.0
  SCAN1 DETECTOR_Z 0.0 0.0 0.0 -240.0 0.0 0.0
  SCAN1 DETECTOR_Y 0.0 0.0 0.0 0.6 0.0 0.0
  SCAN1 DETECTOR_X 0.0 0.0 0.0 -0.5 0.0 0.0
  SCAN1 DETECTOR_PITCH 0.0 0.0 0.0 0.0 0.0 0.0

# category DIFFRN_SCAN_FRAME
loop_
_diffrn_scan_frame.frame_id
_diffrn_scan_frame.frame_number
_diffrn_scan_frame.integration_time
_diffrn_scan_frame.scan_id
_diffrn_scan_frame.date
  FRAME1 1 20.0 SCAN1 1997-12-04T10:23:48

# category DIFFRN_SCAN_FRAME_AXIS
loop_
_diffrn_scan_frame_axis.frame_id
_diffrn_scan_frame_axis.axis_id
_diffrn_scan_frame_axis.angle
_diffrn_scan_frame_axis.displacement
  FRAME1 GONIOMETER_OMEGA 12.0 0.0
  FRAME1 GONIOMETER_KAPPA 23.3 0.0
  FRAME1 GONIOMETER_PHI -165.8 0.0
  FRAME1 DETECTOR_Z 0.0 -240.0
  FRAME1 DETECTOR_Y 0.0 0.6
  FRAME1 DETECTOR_X 0.0 -0.5
  FRAME1 DETECTOR_PITCH 0.0 0.0
```

```
# category AXIS
loop_
_axis.id
_axis.type
_axis.equipment
_axis.depends_on
_axis.vector[1] _axis.vector[2] _axis.vector[3]
_axis.offset[1] _axis.offset[2] _axis.offset[3]
GONIOMETER_OMEGA rotation goniometer . 1 0 0 . . .
GONIOMETER_KAPPA rotation goniometer GONIOMETER_OMEGA
     0.64279 0 0.76604 . . .
GONIOMETER_PHI   rotation goniometer GONIOMETER_KAPPA
     1 0 0 . . .
SOURCE           general source . 0 0 1 . . .
GRAVITY          general gravity . 0 -1 0 . . .
DETECTOR_Z       translation detector . 0 0 1 0 0 0
DETECTOR_Y       translation detector DETECTOR_Z 0 1 0 0 0 0
DETECTOR_X       translation detector DETECTOR_Y 1 0 0 0 0 0
DETECTOR_PITCH   rotation    detector DETECTOR_X 0 1 0 0 0 0
ELEMENT_X        translation detector DETECTOR_PITCH
     1 0 0 172.43 -172.43 0
ELEMENT_Y        translation detector ELEMENT_X
     0 1 0 0 0 0

# category ARRAY_STRUCTURE_LIST
loop_
_array_structure_list.array_id
_array_structure_list.index
_array_structure_list.dimension
_array_structure_list.precedence
_array_structure_list.direction
_array_structure_list.axis_set_id
  ARRAY1 1 2300 1 increasing ELEMENT_X
  ARRAY1 2 2300 2 increasing ELEMENT_Y

# category ARRAY_STRUCTURE_LIST_AXIS
loop_
_array_structure_list_axis.axis_set_id
_array_structure_list_axis.axis_id
_array_structure_list_axis.displacement
_array_structure_list_axis.displacement_increment
  ELEMENT_X ELEMENT_X 0.075 0.150
  ELEMENT_Y ELEMENT_Y 0.075 0.150

# category ARRAY_ELEMENT_SIZE
loop_
_array_element_size.array_id
_array_element_size.index
_array_element_size.size
  ARRAY1 1 150e-6
  ARRAY1 2 150e-6

# category ARRAY_INTENSITIES
loop_
_array_intensities.array_id
_array_intensities.binary_id
_array_intensities.linearity
_array_intensities.gain
_array_intensities.gain_esd
_array_intensities.overload
_array_intensities.undefined_value
  ARRAY1 1 linear 1.15 0.2 240000 0

# category ARRAY_STRUCTURE
loop_
_array_structure.id
_array_structure.encoding_type
_array_structure.compression_type
_array_structure.byte_order
  ARRAY1 "signed 32-bit integer" packed little_endian

# category ARRAY_DATA
loop_
_array_data.array_id
_array_data.binary_id
_array_data.data
ARRAY1 1
;
 --CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BASE64
X-Binary-Size: 3801324
X-Binary-ID: 1
X-Binary-Element-Type: "signed 32-bit integer"
Content-MD5: 071ZFvF+aOcW85IN7usl8A==

AABRAAAAAAAAAAAAAAAAAAAA ...AAZBQSr1sKNBOeOe9HITdMdDUnbq7bg
...
8REo6TtBrxJ1vKqAvx9YDMD6...r/tgssjMIJMXATDsZobL90AEXc4KigE

--CIF-BINARY-FORMAT-SECTION----
;
```

Example 3 – Example 2 revised for a spiral scan (R. M. Sweet, P. J. Ellis & H. J. Bernstein).

A detector is placed 240 mm along the *Z* axis from the goniometer, as in Example 2 above, but in this example the image plate is scanned in a spiral pattern from the outside edge in. The axis for positioning the detector in the *Y* direction depends on the detector *Z* axis. This axis is called DETECTOR_Y. The axis for positioning the detector in the *X* direction depends on the detector *Y* axis (and therefore on the detector *Z* axis). This axis is called DETECTOR_X. This detector may be rotated around the *Y* axis. This rotation axis depends on the three translation axes. It is called DETECTOR_PITCH. A coordinate system is defined on the face of the detector in terms of a coupled rotation axis and radial scan axis to form a spiral scan. The rotation axis is called ELEMENT_ROT and the radial axis is called ELEMENT_RAD. A 150 μm radial pitch and a 75 μm 'constant velocity' angular pitch are assumed. Indexing is carried out first on the rotation axis and the radial axis is made to be dependent on it. The two axes are coupled to form an axis set ELEMENT_SPIRAL.

```
###CBF: VERSION 1.1

data_image_1

# category DIFFRN
_diffrn.id P6MB
_diffrn.crystal_id P6MB_CRYSTAL7

# category DIFFRN_SOURCE
loop_
_diffrn_source.diffrn_id
_diffrn_source.source
_diffrn_source.type
  P6MB synchrotron 'SSRL beamline 9-1'

# category DIFFRN_RADIATION
loop_
_diffrn_radiation.diffrn_id
_diffrn_radiation.wavelength_id
_diffrn_radiation.monochromator
_diffrn_radiation.polarizn_source_ratio
_diffrn_radiation.polarizn_source_norm
_diffrn_radiation.div_x_source
_diffrn_radiation.div_y_source
_diffrn_radiation.div_x_y_source
  P6MB WAVELENGTH1 'Si 111' 0.8 0.0 0.08 0.01 0.00

# category DIFFRN_RADIATION_WAVELENGTH
loop_
_diffrn_radiation_wavelength.id
_diffrn_radiation_wavelength.wavelength
_diffrn_radiation_wavelength.wt
  WAVELENGTH1 0.98 1.0

# category DIFFRN_DETECTOR
loop_
_diffrn_detector.diffrn_id
_diffrn_detector.id
_diffrn_detector.type
_diffrn_detector.number_of_axes
  P6MB MAR345-SN26 'MAR 345' 4

# category DIFFRN_DETECTOR_AXIS
loop_
_diffrn_detector_axis.detector_id
_diffrn_detector_axis.axis_id
  MAR345-SN26 DETECTOR_X
  MAR345-SN26 DETECTOR_Y
  MAR345-SN26 DETECTOR_Z
  MAR345-SN26 DETECTOR_PITCH

# category DIFFRN_DETECTOR_ELEMENT
loop_
_diffrn_detector_element.id
_diffrn_detector_element.detector_id
  ELEMENT1 MAR345-SN26
```

```
# category DIFFRN_DATA_FRAME
loop_
_diffrn_data_frame.id
_diffrn_data_frame.detector_element_id
_diffrn_data_frame.array_id
_diffrn_data_frame.binary_id
  FRAME1 ELEMENT1 ARRAY1 1

# category DIFFRN_MEASUREMENT
loop_
_diffrn_measurement.diffrn_id
_diffrn_measurement.id
_diffrn_measurement.number_of_axes
_diffrn_measurement.method
  P6MB GONIOMETER 3 rotation

# category DIFFRN_MEASUREMENT_AXIS
loop_
_diffrn_measurement_axis.measurement_id
_diffrn_measurement_axis.axis_id
  GONIOMETER GONIOMETER_PHI
  GONIOMETER GONIOMETER_KAPPA
  GONIOMETER GONIOMETER_OMEGA

# category DIFFRN_SCAN
loop_
_diffrn_scan.id
_diffrn_scan.frame_id_start
_diffrn_scan.frame_id_end
_diffrn_scan.frames
  SCAN1 FRAME1 FRAME1 1

# category DIFFRN_SCAN_AXIS
loop_
_diffrn_scan_axis.scan_id
_diffrn_scan_axis.axis_id
_diffrn_scan_axis.angle_start
_diffrn_scan_axis.angle_range
_diffrn_scan_axis.angle_increment
_diffrn_scan_axis.displacement_start
_diffrn_scan_axis.displacement_range
_diffrn_scan_axis.displacement_increment
  SCAN1 GONIOMETER_OMEGA 12.0 1.0 1.0 0.0 0.0 0.0
  SCAN1 GONIOMETER_KAPPA 23.3 0.0 0.0 0.0 0.0 0.0
  SCAN1 GONIOMETER_PHI -165.8 0.0 0.0 0.0 0.0 0.0
  SCAN1 DETECTOR_Z 0.0 0.0 0.0 -240.0 0.0 0.0
  SCAN1 DETECTOR_Y 0.0 0.0 0.0 0.6 0.0 0.0
  SCAN1 DETECTOR_X 0.0 0.0 0.0 -0.5 0.0 0.0
  SCAN1 DETECTOR_PITCH 0.0 0.0 0.0 0.0 0.0 0.0

# category DIFFRN_SCAN_FRAME
loop_
_diffrn_scan_frame.frame_id
_diffrn_scan_frame.frame_number
_diffrn_scan_frame.integration_time
_diffrn_scan_frame.scan_id
_diffrn_scan_frame.date
  FRAME1 1 20.0 SCAN1 1997-12-04T10:23:48

# category DIFFRN_SCAN_FRAME_AXIS
loop_
_diffrn_scan_frame_axis.frame_id
_diffrn_scan_frame_axis.axis_id
_diffrn_scan_frame_axis.angle
_diffrn_scan_frame_axis.displacement
  FRAME1 GONIOMETER_OMEGA 12.0 0.0
  FRAME1 GONIOMETER_KAPPA 23.3 0.0
  FRAME1 GONIOMETER_PHI -165.8 0.0
  FRAME1 DETECTOR_Z 0.0 -240.0
  FRAME1 DETECTOR_Y 0.0 0.6
  FRAME1 DETECTOR_X 0.0 -0.5
  FRAME1 DETECTOR_PITCH 0.0 0.0

# category AXIS
loop_
_axis.id _axis.type _axis.equipment _axis.depends_on
_axis.vector[1] _axis.vector[2] _axis.vector[3]
_axis.offset[1] _axis.offset[2] _axis.offset[3]
GONIOMETER_OMEGA rotation goniometer . 1 0 0 . . .
GONIOMETER_KAPPA rotation goniometer GONIOMETER_OMEGA
     0.64279 0 0.76604 . . .
GONIOMETER_PHI rotation goniometer GONIOMETER_KAPPA
     1 0 0 . . .
SOURCE         general source . 0 0 1 . . .
GRAVITY        general gravity . 0 -1 0 . . .
DETECTOR_Z     translation detector . 0 0 1 0 0 0
DETECTOR_Y     translation detector DETECTOR_Z 0 1 0 0 0 0
DETECTOR_X     translation detector DETECTOR_Y 1 0 0 0 0 0
DETECTOR_PITCH rotation    detector DETECTOR_X 0 1 0 0 0 0
ELEMENT_ROT   translation detector DETECTOR_PITCH 0 0 1 0 0 0
ELEMENT_RAD    translation detector ELEMENT_ROT 0 1 0 0 0 0

# category ARRAY_STRUCTURE_LIST
loop_
_array_structure_list.array_id
_array_structure_list.index
_array_structure_list.dimension
_array_structure_list.precedence
_array_structure_list.direction
_array_structure_list.axis_set_id
  ARRAY1 1 8309900 1 increasing ELEMENT_SPIRAL

# category ARRAY_STRUCTURE_LIST_AXIS
loop_
_array_structure_list_axis.axis_set_id
_array_structure_list_axis.axis_id
_array_structure_list_axis.angle
_array_structure_list_axis.displacement
_array_structure_list_axis.angular_pitch
_array_structure_list_axis.radial_pitch
  ELEMENT_SPIRAL ELEMENT_ROT 0    .  0.075   .
  ELEMENT_SPIRAL ELEMENT_RAD . 172.5  .    -0.150

# category ARRAY_ELEMENT_SIZE
# The actual pixels are 0.075 by 0.150 mm.
# We give the coarser dimension here.
loop_
_array_element_size.array_id
_array_element_size.index
_array_element_size.size
  ARRAY1 1 150e-6

# category ARRAY_INTENSITIES
loop_
_array_intensities.array_id
_array_intensities.binary_id
_array_intensities.linearity
_array_intensities.gain
_array_intensities.gain_esd
_array_intensities.overload
_array_intensities.undefined_value
  ARRAY1 1 linear 1.15 0.2 240000 0

# category ARRAY_STRUCTURE
loop_
_array_structure.id
_array_structure.encoding_type
_array_structure.compression_type
_array_structure.byte_order
  ARRAY1 "signed 32-bit integer" packed little_endian

# category ARRAY_DATA
loop_
_array_data.array_id
_array_data.binary_id
_array_data.data
 ARRAY1 1
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BASE64
X-Binary-Size: 3801324
X-Binary-ID: 1
X-Binary-Element-Type: "signed 32-bit integer"
Content-MD5: 07lZFvF+aOcW85IN7usl8A==

AABRAAAAAAAAAAAAAAAAAAAA ...AAZBQSr1sKNBOeOe9HITdMdDUnbq7bg
 ...
8REo6TtBrxJ1vKqAvx9YDMD6...r/tgssjMIJMXATDsZobL90AEXc4KigE

--CIF-BINARY-FORMAT-SECTION----
;
```

_diffrn_scan.date_end (*yyyy-mm-dd*)
The date and time of the end of the scan. Note that this may be an estimate generated during the scan, before the precise time of the end of the scan is known.
[diffrn_scan]

_diffrn_scan.date_start (*yyyy-mm-dd*)
The date and time of the start of the scan.
[diffrn_scan]

***_diffrn_scan.frame_id_end** (*code*)
The value of this data item is the identifier of the last frame in the scan. This item is a pointer to **_diffrn_data_frame.id** in the DIFFRN_DATA_FRAME category.
[diffrn_scan]

***_diffrn_scan.frame_id_start** (*code*)
The value of this data item is the identifier of the first frame in the scan. This item is a pointer to **_diffrn_data_frame.id** in the DIFFRN_DATA_FRAME category.
[diffrn_scan]

_diffrn_scan.frames (*int*)
The value of this data item is the number of frames in the scan.
The permitted range is $[1, \infty)$. **[diffrn_scan]**

***_diffrn_scan.id** (*code*)
The value of **_diffrn_scan.id** uniquely identifies each scan. The identifier is used to tie together all the information about the scan.
The following item(s) have an equivalent role in their respective categories:
_diffrn_scan_axis.scan_id,
_diffrn_scan_frame.scan_id. **[diffrn_scan]**

_diffrn_scan.integration_time (*float*)
Approximate average time in seconds to integrate each step of the scan. The precise time for integration of each particular step must be provided in **_diffrn_scan_frame.integration_time**, even if all steps have the same integration time.
The permitted range is $[0.0, \infty)$. **[diffrn_scan]**

DIFFRN_SCAN_AXIS

Data items in the DIFFRN_SCAN_AXIS category describe the settings of axes for particular scans. Unspecified axes are assumed to be at their zero points.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_scan_axis.scan_id**
_diffrn_scan_axis.axis_id

_diffrn_scan_axis.angle_increment (*float*)
The increment for each step for the specified axis in degrees. In general, this will agree with **_diffrn_scan_frame_axis.angle_increment**. The sum of the values of **_diffrn_scan_frame_axis.angle** and **_diffrn_scan_frame_axis.angle_increment** is the angular setting of the axis at the end of the integration time for a given frame. If the individual frame values vary, then the value of **_diffrn_scan_axis.angle_increment** will be representative of the ensemble of values of **_diffrn_scan_frame_axis.angle_increment** (*e.g.* the mean).
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

_diffrn_scan_axis.angle_range (*float*)
The range from the starting position for the specified axis in degrees.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

_diffrn_scan_axis.angle_rstrt_incr (*float*)
The increment after each step for the specified axis in degrees. In general, this will agree with **_diffrn_scan_frame_axis.angle_rstrt_incr**. The sum of the values of **_diffrn_scan_frame_axis.angle**, **_diffrn_scan_frame_axis.angle_increment** and **_diffrn_scan_frame_axis.angle_rstrt_incr** is the angular setting of the axis at the start of the integration time for the next frame relative to a given frame and should equal **_diffrn_scan_frame_axis.angle** for this next frame. If the individual frame values vary, then the value of **_diffrn_scan_axis.angle_rstrt_incr** will be representative of the ensemble of values of **_diffrn_scan_frame_axis.angle_rstrt_incr** (*e.g.* the mean).
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

_diffrn_scan_axis.angle_start (*float*)
The starting position for the specified axis in degrees.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

***_diffrn_scan_axis.axis_id** (*code*)
The value of this data item is the identifier of one of the axes for the scan for which settings are being specified. Multiple axes may be specified for the same value of **_diffrn_scan.id**. This item is a pointer to **_axis.id** in the AXIS category.
[diffrn_scan_axis]

_diffrn_scan_axis.displacement_increment (*float*)
The increment for each step for the specified axis in millimetres. In general, this will agree with **_diffrn_scan_frame_axis.displacement_increment**. The sum of the values of **_diffrn_scan_frame_axis.displacement** and **_diffrn_scan_frame_axis.displacement_increment** is the angular setting of the axis at the end of the integration time for a given frame. If the individual frame values vary, then the value of **_diffrn_scan_axis.displacement_increment** will be representative of the ensemble of values of **_diffrn_scan_frame_axis.displacement_increment** (*e.g.* the mean).
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

_diffrn_scan_axis.displacement_range (*float*)
The range from the starting position for the specified axis in millimetres.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

_diffrn_scan_axis.displacement_rstrt_incr (*float*)
The increment for each step for the specified axis in millimetres. In general, this will agree with **_diffrn_scan_frame_axis.displacement_rstrt_incr**. The sum of the values of **_diffrn_scan_frame_axis.displacement**, **_diffrn_scan_frame_axis.displacement_increment** and **_diffrn_scan_frame_axis.displacement_rstrt_incr** is the angular setting of the axis at the start of the integration time for the next frame relative to a given frame and should equal **_diffrn_scan_frame_axis.displacement** for this next frame. If the individual frame values vary, then the value of **_diffrn_scan_axis.displacement_rstrt_incr** will be representative of the ensemble of values of **_diffrn_scan_frame_axis.displacement_rstrt_incr** (*e.g.* the mean).
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

_diffrn_scan_axis.displacement_start (*float*)
The starting position for the specified axis in millimetres.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_axis]**

***_diffrn_scan_axis.scan_id** (*code*)
The value of this data item is the identifier of the scan for which axis settings are being specified. Multiple axes may be specified for the same value of **_diffrn_scan.id**. This item is a pointer to **_diffrn_scan.id** in the DIFFRN_SCAN category.
[diffrn_scan_axis]

DIFFRN_SCAN_FRAME

Data items in the DIFFRN_SCAN_FRAME category describe the relationships of particular frames to scans.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_scan_frame.scan_id**
_diffrn_scan_frame.frame_id

_diffrn_scan_frame.date (*yyyy-mm-dd*)
The date and time of the start of the frame being scanned.
[diffrn_scan_frame]

***_diffrn_scan_frame.frame_id** (*code*)
The value of this data item is the identifier of the frame being examined. This item is a pointer to **_diffrn_data_frame.id** in the DIFFRN_DATA_FRAME category.
[diffrn_scan_frame]

_diffrn_scan_frame.frame_number (*int*)
The value of this data item is the number of the frame within the scan, starting with 1. It is not necessarily the same as the value of **_diffrn_scan_frame.frame_id**, but it may be.
The permitted range is $[0, \infty)$. **[diffrn_scan_frame]**

***_diffrn_scan_frame.integration_time** (*float*)
The time in seconds to integrate this step of the scan. This should be the precise time of integration of each particular frame. The value of this data item should be given explicitly for each frame and not inferred from the value of **_diffrn_scan.integration_time**.
The permitted range is $[0.0, \infty)$. **[diffrn_scan_frame]**

***_diffrn_scan_frame.scan_id** (*code*)
The value of **_diffrn_scan_frame.scan_id** identifies the scan containing this frame. This item is a pointer to **_diffrn_scan.id** in the DIFFRN_SCAN category.
[diffrn_scan_frame]

DIFFRN_SCAN_FRAME_AXIS

Data items in the DIFFRN_SCAN_FRAME_AXIS category describe the settings of axes for particular frames. Unspecified axes are assumed to be at their zero points. If, for any given frame, nonzero values apply for any of the data items in this category, those values should be given explicitly in this category and not simply inferred from values in DIFFRN_SCAN_AXIS.
Category group(s): **inclusive_group**
diffrn_group
Category key(s): **_diffrn_scan_frame_axis.frame_id**
_diffrn_scan_frame_axis.axis_id

_diffrn_scan_frame_axis.angle (*float*)
The setting of the specified axis in degrees for this frame. This is the setting at the start of the integration time.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_frame_axis]**

_diffrn_scan_frame_axis.angle_increment (*float*)
The increment for this frame for the angular setting of the specified axis in degrees. The sum of the values of **_diffrn_scan_frame_axis.angle** and **_diffrn_scan_frame_axis.angle_increment** is the angular setting of the axis at the end of the integration time for this frame.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_frame_axis]**

_diffrn_scan_frame_axis.angle_rstrt_incr (*float*)
The increment after this frame for the angular setting of the specified axis in degrees. The sum of the values of **_diffrn_scan_frame_axis.angle**, **_diffrn_scan_frame_axis.angle_increment** and **_diffrn_scan_frame_axis.angle_rstrt_incr** is the angular setting of the axis at the start of the integration time for the next frame and should equal **_diffrn_scan_frame_axis.angle** for this next frame.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_frame_axis]**

***_diffrn_scan_frame_axis.axis_id** (*code*)
The value of this data item is the identifier of one of the axes for the frame for which settings are being specified. Multiple axes may be specified for the same value of **_diffrn_scan_frame.frame_id**. This item is a pointer to **_axis.id** in the AXIS category.
[diffrn_scan_frame_axis]

_diffrn_scan_frame_axis.displacement (*float*)
The setting of the specified axis in millimetres for this frame. This is the setting at the start of the integration time.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_frame_axis]**

_diffrn_scan_frame_axis.displacement_increment (*float*)
The increment for this frame for the displacement setting of the specified axis in millimetres. The sum of the values of **_diffrn_scan_frame_axis.displacement** and **_diffrn_scan_frame_axis.displacement_increment** is the angular setting of the axis at the end of the integration time for this frame.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_frame_axis]**

_diffrn_scan_frame_axis.displacement_rstrt_incr (*float*)
The increment for this frame for the displacement setting of the specified axis in millimetres. The sum of the values of **_diffrn_scan_frame_axis.displacement**, **_diffrn_scan_frame_axis.displacement_increment** and **_diffrn_scan_frame_axis.displacement_rstrt_incr** is the angular setting of the axis at the start of the integration time for the next frame and should equal **_diffrn_scan_frame_axis.displacement** for this next frame.
Where no value is given, the assumed value is '0.0'. **[diffrn_scan_frame_axis]**

***_diffrn_scan_frame_axis.frame_id** (*code*)
The value of this data item is the identifier of the frame for which axis settings are being specified. Multiple axes may be specified for the same value of **_diffrn_scan_frame.frame_id**. This item is a pointer to **_diffrn_data_frame.id** in the DIFFRN_DATA_FRAME category.
[diffrn_scan_frame_axis]

4.7. Symmetry dictionary (symCIF)

BY I. D. BROWN

This is version 1.0.1 of the symmetry CIF dictionary (symCIF), which gives a more complete description of symmetry than was included in the original core CIF dictionary. A detailed commentary on the philosophy behind the dictionary and its use may be found in Chapter 3.8.

SPACE_GROUP

Contains all the data items that refer to the space group as a whole, such as its name, Laue group *etc.* It may be looped, for example in a list of space groups and their properties. Space-group types are identified by their number as listed in *International Tables for Crystallography* Volume A, or by their Schoenflies symbol. Specific settings of the space groups can be identified by their Hall symbol, by specifying their symmetry operations or generators, or by giving the transformation that relates the specific setting to the reference setting based on *International Tables* Volume A and stored in this dictionary. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but several different Hermann–Mauguin symbols may refer to the same space-group type. A Hermann–Mauguin symbol contains information on the choice of the basis, but not on the choice of origin.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Mandatory category.

Category key(s): **`_space_group.id`**

Example 1 – description of the C2/c space group, No. 15 in International Tables for Crystallography Volume A.

```
_space_group.id                    1
_space_group.name_H-M_ref          'C 2/c'
_space_group.name_Schoenflies      C2h.6
_space_group.IT_number             15
_space_group.name_Hall             '-C 2yc'
_space_group.Bravais_type          mS
_space_group.Laue_class            2/m
_space_group.crystal_system        monoclinic
_space_group.centring_type         C
_space_group.Patterson_name_H-M    'C 2/m'
```

`_space_group.Bravais_type` (*char*)

The symbol denoting the lattice type (Bravais type) to which the translational subgroup (vector lattice) of the space group belongs. It consists of a lower-case letter indicating the crystal system followed by an upper-case letter indicating the lattice centring. The setting-independent symbol *mS* replaces the setting-dependent symbols *mB* and *mC*, and the setting-independent symbol *oS* replaces the setting-dependent symbols *oA*, *oB* and *oC*.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed., p. 15. Dordrecht: Kluwer Academic Publishers.

The data value must be one of the following:

`aP`
`mP mS`
`oP oS oI oF`
`tP tI`
`hP hR`
`cP cI cF`

Example: '`aP`' (triclinic (anorthic) primitive lattice). **`[space_group]`**

`_space_group.IT_coordinate_system_code` (*char*)

A qualifier taken from the enumeration list identifying which setting in *International Tables for Crystallography* Volume A (2002) (*IT*) is used. See *IT* Table 4.3.2.1, Section 2.2.16, Table 2.2.16.1, Section 2.2.16.1 and Fig. 2.2.6.4. This item is not computer-interpretable and cannot be used to define the coordinate system. Use **`_space_group.transform_`*** instead.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

The data value must be one of the following:

`b1`	monoclinic unique axis *b*, cell choice 1, **abc**
`b2`	monoclinic unique axis *b*, cell choice 2, **abc**
`b3`	monoclinic unique axis *b*, cell choice 3, **abc**
`-b1`	monoclinic unique axis *b*, cell choice 1, **c$\bar{b}$a**
`-b2`	monoclinic unique axis *b*, cell choice 2, **c$\bar{b}$a**
`-b3`	monoclinic unique axis *b*, cell choice 3, **c$\bar{b}$a**
`c1`	monoclinic unique axis *c*, cell choice 1, **abc**
`c2`	monoclinic unique axis *c*, cell choice 2, **abc**
`c3`	monoclinic unique axis *c*, cell choice 3, **abc**
`-c1`	monoclinic unique axis *c*, cell choice 1, **ba$\bar{c}$**
`-c2`	monoclinic unique axis *c*, cell choice 2, **ba$\bar{c}$**
`-c3`	monoclinic unique axis *c*, cell choice 3, **ba$\bar{c}$**
`a1`	monoclinic unique axis *a*, cell choice 1, **abc**
`a2`	monoclinic unique axis *a*, cell choice 2, **abc**
`a3`	monoclinic unique axis *a*, cell choice 3, **abc**
`-a1`	monoclinic unique axis *a*, cell choice 1, **$\bar{a}$cb**
`-a2`	monoclinic unique axis *a*, cell choice 2, **$\bar{a}$cb**
`-a3`	monoclinic unique axis *a*, cell choice 3, **$\bar{a}$cb**
`abc`	orthorhombic
`ba-c`	orthorhombic
`cab`	orthorhombic
`-cba`	orthorhombic
`bca`	orthorhombic
`a-cb`	orthorhombic
`1abc`	orthorhombic origin choice 1
`1ba-c`	orthorhombic origin choice 1
`1cab`	orthorhombic origin choice 1
`1-cba`	orthorhombic origin choice 1
`1bca`	orthorhombic origin choice 1
`1a-cb`	orthorhombic origin choice 1
`2abc`	orthorhombic origin choice 2
`2ba-c`	orthorhombic origin choice 2
`2cab`	orthorhombic origin choice 2
`2-cba`	orthorhombic origin choice 2
`2bca`	orthorhombic origin choice 2
`2a-cb`	orthorhombic origin choice 2
`1`	tetragonal or cubic origin choice 1
`2`	tetragonal or cubic origin choice 2
`h`	trigonal using hexagonal axes
`r`	trigonal using rhombohedral axes

`[space_group]`

Affiliation: I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1.

_space_group.IT_number (*numb*)

_symmetry_Int_Tables_number (*cif_core.dic 1.0*)

The number as assigned in *International Tables for Crystallography* Volume A, specifying the proper affine class (*i.e.* the orientation-preserving affine class) of space groups (crystallographic space-group type) to which the space group belongs. This number defines the space-group type but not the coordinate system in which it is expressed.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

The permitted range is (1, 230). **[space_group]**

_space_group.Laue_class (*char*)

The Hermann–Mauguin symbol of the geometric crystal class of the point group of the space group where a centre of inversion is added if not already present.

The data value must be one of the following:

```
-1
2/m    mmm
4/m    4/mmm
-3     -3m
6/m    6/mmm
m-3    m-3m
```
[space_group]

_space_group.Patterson_name_H-M (*char*)

The Hermann–Mauguin symbol of the type of that centrosymmetric symmorphic space group to which the Patterson function belongs; see Table 2.2.5.1 in *International Tables for Crystallography* Volume A (2002). A space separates each symbol referring to different axes. Underscores may replace the spaces, but this use is discouraged. Subscripts should appear without special symbols. Bars should be given as negative signs before the number to which they apply.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed., Table 2.2.5.1. Dordrecht: Kluwer Academic Publishers.

Examples: 'P -1', 'P 2/m', 'C 2/m', 'P m m m', 'C m m m', 'I m m m', 'F m m m', 'P 4/m', 'I 4/m', 'P 4/m m m', 'I 4/m m m', 'P -3', 'R -3', 'P -3 m 1', 'R -3 m', 'P -3 1 m', 'P 6/m', 'P 6/m m m', 'P m -3', 'I m -3', 'F m -3', 'P m -3 m', 'I m -3 m', 'F m -3 m'. **[space_group]**

_space_group.centring_type (*char*)

Symbol for the lattice centring. This symbol may be dependent on the coordinate system chosen.

The data value must be one of the following:

P	primitive no centring
A	*A*-face centred (0, 1/2, 1/2)
B	*B*-face centred (1/2, 0, 1/2)
C	*C*-face centred (1/2, 1/2, 0)
F	all faces centred (0, 1/2, 1/2), (1/2, 0, 1/2), (1/2, 1/2, 0)
I	body centred (1/2, 1/2, 1/2)
R	rhombohedral obverse centred (2/3, 1/3, 1/3), (1/3, 2/3, 2/3)
Rrev	rhombohedral reverse centred (1/3, 2/3, 1/3), (2/3, 1/3, 2/3)
H	hexagonal centred (2/3, 1/3, 0), (1/3, 2/3, 0)

[space_group]

_space_group.crystal_system (*char*)

_symmetry_cell_setting (*cif_core.dic 1.0*)

The name of the system of geometric crystal classes of space groups (crystal system) to which the space group belongs. Note that crystals with the *hR* lattice type belong to the trigonal system.

The data value must be one of the following:

```
triclinic
monoclinic
orthorhombic
tetragonal
trigonal
hexagonal
cubic
```
[space_group]

***_space_group.id** (*char*)

This is an identifier needed if **_space_group.*** items are looped.

The following item(s) have an equivalent role in their respective categories:

_space_group_symop.sg_id,

_space_group_Wyckoff.sg_id. **[space_group]**

_space_group.name_H-M_alt (*char*)

_symmetry_space_group_name_H-M (*cif_core.dic 1.0*)

_space_group.name_H-M_alt allows for an alternative Hermann–Mauguin symbol to be given. The way in which this item is used is determined by the user and should be described in the item **_space_group.name_H-M_alt_description**. It may, for example, be used to give one of the extended Hermann–Mauguin symbols given in Table 4.3.2.1 of *International Tables for Crystallography* Volume A (2002) or a full Hermann–Mauguin symbol for an unconventional setting. Each component of the space-group name is separated by a space or an underscore character. The use of a space is strongly recommended. The underscore is only retained because it was used in older CIFs. It should not be used in new CIFs. Subscripts should appear without special symbols. Bars should be given as negative signs before the numbers to which they apply. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely, but a given space-group type may be described by more than one Hermann–Mauguin symbol. The space-group type is best described using **_space_group.IT_number** or **_space_group.name_Schoenflies**. The Hermann–Mauguin symbol may contain information on the choice of basis but does not contain information on the choice of origin. To define the setting uniquely, use **_space_group.name_Hall**, list the symmetry operations or generators, or give the transformation that relates the setting to the reference setting defined in this dictionary under **_space_group.reference_setting**.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Related items: **_space_group.name_H-M_ref** (alternate), **_space_group.name_H-M_full** (alternate) .

Example:

```
; loop_
  _space_group.name_H-M_alt
  _space_group.name_H-M_alt_description
  'C m c m(b n n)'
  'Extended Hermann-Mauguin symbol'
  'C 2/c 2/m 21/m'
  'Full unconventional Hermann-Mauguin symbol'
  'A m a m'
  'Hermann-Mauguin symbol corresponding to setting used'
;  (Three examples for space group No. 63.)
```
[space_group]

_space_group.name_H-M_alt_description (*char*)

A free-text description of the code appearing in **_space_group.name_H-M_alt**.

[space_group]

_space_group.name_H-M_full (*char*)

symmetry.space_group_name_H-M (*cif_mm.dic 1.0.0*)

The full international Hermann–Mauguin space-group symbol as defined in Section 2.2.3 and given as the second item of the second line of each of the space-group tables of Part 7 of *International Tables for Crystallography* Volume A (2002). Each component of the space-group name is separated by a space or an underscore character. The use of a space is strongly recommended. The underscore is only retained because it was used in old CIFs. It should not be used in new CIFs. Subscripts should appear without special symbols. Bars should be given as negative signs before the numbers to which they apply. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but a given

space-group type may be described by more than one Hermann–Mauguin symbol. The space-group type is best described using **_space_group.IT_number** or **_space_group.name_Schoenflies**. The full international Hermann–Mauguin symbol contains information about the choice of basis for monoclinic and orthorhombic space groups but does not give information about the choice of origin. To define the setting uniquely use **_space_group.name_Hall**, list the symmetry operations or generators, or give the transformation relating the setting used to the reference setting defined in this dictionary under **_space_group.reference_setting**.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Related items: **_space_group.name_H-M_ref** (alternate), **_space_group.name_H-M_alt** (alternate) .

Example: '`P 21/n 21/m 21/a`' (full symbol for *Pnma*). **[space_group]**

_space_group.name_H-M_ref (*char*)

The short international Hermann–Mauguin space-group symbol as defined in Section 2.2.3 and given as the first item of each space-group table in Part 7 of *International Tables for Crystallography* Volume A (2002). Each component of the space-group name is separated by a space or an underscore character. The use of a space is strongly recommended. The underscore is only retained because it was used in old CIFs. It should not be used in new CIFs. Subscripts should appear without special symbols. Bars should be given as negative signs before the numbers to which they apply. The short international Hermann–Mauguin symbol determines the space-group type uniquely. However, the space-group type is better described using **_space_group.IT_number** or **_space_group.name_Schoenflies**. The short international Hermann–Mauguin symbol contains no information on the choice of basis or origin. To define the setting uniquely use **_space_group.name_Hall**, list the symmetry operations or generators, or give the transformation that relates the setting to the reference setting defined in this dictionary under **_space_group.reference_setting**. **_space_group.name_H-M_alt** may be used to give the Hermann–Mauguin symbol corresponding to the setting used. In the enumeration list below, each possible value is identified by space-group number and Schoenflies symbol.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Related items: **_space_group.name_H-M_full** (alternate), **_space_group.name_H-M_alt** (alternate).

The data value must be one of the following:

'P 1'	1 C_1^1	'P -1'	2 C_i^1
'P 2'	3 C_2^1	'P 21'	4 C_2^2
'C 2'	5 C_2^3	'P m'	6 C_s^1
'P c'	7 C_s^2	'C m'	8 C_s^3
'C c'	9 C_s^4	'P 2/m'	10 C_{2h}^1
'P 21/m'	11 C_{2h}^2	'C 2/m'	12 C_{2h}^3
'P 2/c'	13 C_{2h}^4	'P 21/c'	14 C_{2h}^5
'C 2/c'	15 C_{2h}^6	'P 2 2 2'	16 D_2^1
'P 2 2 21'	17 D_2^2	'P 21 21 2'	18 D_2^3
'P 21 21 21'	19 D_2^4	'C 2 2 21'	20 D_2^5
'C 2 2 2'	21 D_2^6	'F 2 2 2'	22 D_2^7
'I 2 2 2'	23 D_2^8	'I 21 21 21'	24 D_2^9
'P m m 2'	25 C_{2v}^1	'P m c 21'	26 C_{2v}^2
'P c c 2'	27 C_{2v}^3	'P m a 2'	28 C_{2v}^4
'P c a 21'	29 C_{2v}^5	'P n c 2'	30 C_{2v}^6
'P m n 21'	31 C_{2v}^7	'P b a 2'	32 C_{2v}^8
'P n a 21'	33 C_{2v}^9	'P n n 2'	34 C_{2v}^{10}
'C m m 2'	35 C_{2v}^{11}	'C m c 21'	36 C_{2v}^{12}
'C c c 2'	37 C_{2v}^{13}	'A m m 2'	38 C_{2v}^{14}
'A e m 2'	39 C_{2v}^{15}	'A m a 2'	40 C_{2v}^{16}
'A e a 2'	41 C_{2v}^{17}	'F m m 2'	42 C_{2v}^{18}
'F d d 2'	43 C_{2v}^{19}	'I m m 2'	44 C_{2v}^{20}
'I b a 2'	45 C_{2v}^{21}	'I m a 2'	46 C_{2v}^{22}
'P m m m'	47 D_{2h}^1	'P n n n'	48 D_{2h}^2
'P c c m'	49 D_{2h}^3	'P b a n'	50 D_{2h}^4
'P m m a'	51 D_{2h}^5	'P n n a'	52 D_{2h}^6
'P m n a'	53 D_{2h}^7	'P c c a'	54 D_{2h}^8
'P b a m'	55 D_{2h}^9	'P c c n'	56 D_{2h}^{10}
'P b c m'	57 D_{2h}^{11}	'P n n m'	58 D_{2h}^{12}
'P m m n'	59 D_{2h}^{13}	'P b c n'	60 D_{2h}^{14}
'P b c a'	61 D_{2h}^{15}	'P n m a'	62 D_{2h}^{16}
'C m c m'	63 D_{2h}^{17}	'C m c e'	64 D_{2h}^{18}
'C m m m'	65 D_{2h}^{19}	'C c c m'	66 D_{2h}^{20}
'C m m e'	67 D_{2h}^{21}	'C c c e'	68 D_{2h}^{22}
'F m m m'	69 D_{2h}^{23}	'F d d d'	70 D_{2h}^{24}
'I m m m'	71 D_{2h}^{25}	'I b a m'	72 D_{2h}^{26}
'I b c a'	73 D_{2h}^{27}	'I m m a'	74 D_{2h}^{28}
'P 4'	75 C_4^1	'P 41'	76 C_4^2
'P 42'	77 C_4^3	'P 43'	78 C_4^4
'I 4'	79 C_4^5	'I 41'	80 C_4^6
'P -4'	81 S_4^1	'I -4'	82 S_4^2
'P 4/m'	83 C_{4h}^1	'P 42/m'	84 C_{4h}^2
'P 4/n'	85 C_{4h}^3	'P 42/n'	86 C_{4h}^4
'I 4/m'	87 C_{4h}^5	'I 41/a'	88 C_{4h}^6
'P 4 2 2'	89 D_4^1	'P 4 21 2'	90 D_4^2
'P 41 2 2'	91 D_4^3	'P 41 21 2'	92 D_4^4
'P 42 2 2'	93 D_4^5	'P 42 21 2'	94 D_4^6
'P 43 2 2'	95 D_4^7	'P 43 21 2'	96 D_4^8
'I 4 2 2'	97 D_4^9	'I 41 2 2'	98 D_4^{10}
'P 4 m m'	99 C_{4v}^1	'P 4 b m'	100 C_{4v}^2
'P 42 c m'	101 C_{4v}^3	'P 42 n m'	102 C_{4v}^4
'P 4 c c'	103 C_{4v}^5	'P 4 n c'	104 C_{4v}^6
'P 42 m c'	105 C_{4v}^7	'P 42 b c'	106 C_{4v}^8
'I 4 m m'	107 C_{4v}^9	'I 4 c m'	108 C_{4v}^{10}
'I 41 m d'	109 C_{4v}^{11}	'I 41 c d'	110 C_{4v}^{12}
'P -4 2 m'	111 D_{2d}^1	'P -4 2 c'	112 D_{2d}^2
'P -4 21 m'	113 D_{2d}^3	'P -4 21 c'	114 D_{2d}^4
'P -4 m 2'	115 D_{2d}^5	'P -4 c 2'	116 D_{2d}^6
'P -4 b 2'	117 D_{2d}^7	'P -4 n 2'	118 D_{2d}^8
'I -4 m 2'	119 D_{2d}^9	'I -4 c 2'	120 D_{2d}^{10}
'I -4 2 m'	121 D_{2d}^{11}	'I -4 2 d'	122 D_{2d}^{12}
'P 4/m m m'	123 D_{4h}^1	'P 4/m c c'	124 D_{4h}^2
'P 4/n b m'	125 D_{4h}^3	'P 4/n n c'	126 D_{4h}^4
'P 4/m b m'	127 D_{4h}^5	'P 4/m n c'	128 D_{4h}^6
'P 4/n m m'	129 D_{4h}^7	'P 4/n c c'	130 D_{4h}^8
'P 42/m m c'	131 D_{4h}^9	'P 42/m c m'	132 D_{4h}^{10}
'P 42/n b c'	133 D_{4h}^{11}	'P 42/n n m'	134 D_{4h}^{12}
'P 42/m b c'	135 D_{4h}^{13}	'P 42/m n m'	136 D_{4h}^{14}
'P 42/n m c'	137 D_{4h}^{15}	'P 42/n c m'	138 D_{4h}^{16}
'I 4/m m m'	139 D_{4h}^{17}	'I 4/m c m'	140 D_{4h}^{18}
'I 41/a m d'	141 D_{4h}^{19}	'I 41/a c d'	142 D_{4h}^{20}
'P 3'	143 C_3^1	'P 31'	144 C_3^2
'P 32'	145 C_3^3	'R 3'	146 C_3^4
'P -3'	147 C_{3i}^1	'R -3'	148 C_{3i}^2
'P 3 1 2'	149 D_3^1	'P 3 2 1'	150 D_3^2
'P 31 1 2'	151 D_3^3	'P 31 2 1'	152 D_3^4

'P 32 1 2'	153	D_3^5	'P 32 2 1'	154	D_3^6
'R 3 2'	155	D_3^7	'P 3 m 1'	156	C_{3v}^1
'P 3 1 m'	157	C_{3v}^2	'P 3 c 1'	158	C_{3v}^3
'P 3 1 c'	159	C_{3v}^4	'R 3 m'	160	C_{3v}^5
'R 3 c'	161	C_{3v}^6	'P -3 1 m'	162	D_{3d}^1
'P -3 1 c'	163	D_{3d}^2	'P -3 m 1'	164	D_{3d}^3
'P -3 c 1'	165	D_{3d}^4	'R -3 m'	166	D_{3d}^5
'R -3 c'	167	D_{3d}^6	'P 6'	168	C_6^1
'P 61'	169	C_6^2	'P 65'	170	C_6^3
'P 62'	171	C_6^4	'P 64'	172	C_6^5
'P 63'	173	C_6^6	'P -6'	174	C_{3h}^1
'P 6/m '	175	C_{6h}^1	'P 63/m'	176	C_{6h}^2
'P 6 2 2'	177	D_6^1	'P 61 2 2'	178	D_6^2
'P 65 2 2'	179	D_6^3	'P 62 2 2'	180	D_6^4
'P 64 2 2'	181	D_6^5	'P 63 2 2'	182	D_6^6
'P 6 m m'	183	C_{6v}^1	'P 6 c c'	184	C_{6v}^2
'P 63 c m'	185	C_{6v}^3	'P 63 m c'	186	C_{6v}^4
'P -6 m 2'	187	D_{3h}^1	'P -6 c 2'	188	D_{3h}^2
'P -6 2 m'	189	D_{3h}^3	'P -6 2 c'	190	D_{3h}^4
'P 6/m m m'	191	D_{6h}^1	'P 6/m c c'	192	D_{6h}^2
'P 63/m c m'	193	D_{6h}^3	'P 63/m m c'	194	D_{6h}^4
'P 2 3'	195	T^1	'F 2 3'	196	T^2
'I 2 3'	197	T^3	'P 21 3'	198	T^4
'I 21 3'	199	T^5	'P m -3'	200	T_h^1
'P n -3'	201	T_h^2	'F m -3'	202	T_h^3
'F d -3'	203	T_h^4	'I m -3'	204	T_h^5
'P a -3'	205	T_h^6	'I a -3'	206	T_h^7
'P 4 3 2'	207	O^1	'P 42 3 2'	208	O^2
'F 4 3 2'	209	O^3	'F 41 3 2'	210	O^4
'I 4 3 2'	211	O^5	'P 43 3 2'	212	O^6
'P 41 3 2'	213	O^7	'I 41 3 2'	214	O^8
'P -4 3 m'	215	T_d^1	'F -4 3 m'	216	T_d^2
'I -4 3 m'	217	T_d^3	'P -4 3 n'	218	T_d^4
'F -4 3 c'	219	T_d^5	'I -4 3 d'	220	T_d^6
'P m -3 m'	221	O_h^1	'P n -3 n'	222	O_h^2
'P m -3 n'	223	O_h^3	'P n -3 m'	224	O_h^4
'F m -3 m'	225	O_h^5	'F m -3 c'	226	O_h^6
'F d -3 m'	227	O_h^7	'F d -3 c'	228	O_h^8
'I m -3 m'	229	O_h^9	'I a -3 d'	230	O_h^{10}

Examples: '`P 21/c`', '`P m n a`', '`P -1`', '`F m -3 m`', '`P 63/m m m`'.

[space_group]

_space_group.name_Hall (*char*)

_symmetry_space_group_name_Hall (*cif_core.dic 1.0*)

Space-group symbol defined by Hall. **_space_group.name_Hall** uniquely defines the space group and its reference to a particular coordinate system. Each component of the space-group name is separated by a space or an underscore character. The use of a space is strongly recommended. The underscore is only retained because it was used in old CIFs. It should not be used in new CIFs.

References: Hall, S. R. (1981). *Acta Cryst.* A**37**, 517–525; erratum (1981), A**37**, 921. *International Tables for Crystallography* (2001). Volume B, *Reciprocal space*, edited by U. Shmueli, 2nd ed., Appendix 1.4.2. Dordrecht: Kluwer Academic Publishers.

Examples: '`P 2c -2ac`' (equivalent to $Pca2_1$), '`-I 4bd 2ab 3`' (equivalent to $Ia3d$).

[space_group]

_space_group.name_Schoenflies (*char*)

The Schoenflies symbol as listed in *International Tables for Crystallography* Volume A denoting the proper affine class (*i.e.* orientation-preserving affine class) of space groups (space-group type) to which the space group belongs. This symbol defines the space-group type independently of the coordinate system in which the space group is expressed. The symbol is given with a period, '.', separating the Schoenflies point group and the superscript.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

The data value must be one of the following:

C1.1	Ci.1	C2.1	C2.2	C2.3	Cs.1	Cs.2
Cs.3	Cs.4	C2h.1	C2h.2	C2h.3	C2h.4	C2h.5
C2h.6	D2.1	D2.2	D2.3	D2.4	D2.5	D2.6
D2.7	D2.8	D2.9	C2v.1	C2v.2	C2v.3	C2v.4
C2v.5	C2v.6	C2v.7	C2v.8	C2v.9	C2v.10	C2v.11
C2v.12	C2v.13	C2v.14	C2v.15	C2v.16	C2v.17	C2v.18
C2v.19	C2v.20	C2v.21	C2v.22	D2h.1	D2h.2	D2h.3
D2h.4	D2h.5	D2h.6	D2h.7	D2h.8	D2h.9	D2h.10
D2h.11	D2h.12	D2h.13	D2h.14	D2h.15	D2h.16	D2h.17
D2h.18	D2h.19	D2h.20	D2h.21	D2h.22	D2h.23	D2h.24
D2h.25	D2h.26	D2h.27	D2h.28	C4.1	C4.2	C4.3
C4.4	C4.5	C4.6	S4.1	S4.2	C4h.1	C4h.2
C4h.3	C4h.4	C4h.5	C4h.6	D4.1	D4.2	D4.3
D4.4	D4.5	D4.6	D4.7	D4.8	D4.9	D4.10
C4v.1	C4v.2	C4v.3	C4v.4	C4v.5	C4v.6	C4v.7
C4v.8	C4v.9	C4v.10	C4v.11	C4v.12	D2d.1	D2d.2
D2d.3	D2d.4	D2d.5	D2d.6	D2d.7	D2d.8	D2d.9
D2d.10	D2d.11	D2d.12	D4h.1	D4h.2	D4h.3	D4h.4
D4h.5	D4h.6	D4h.7	D4h.8	D4h.9	D4h.10	D4h.11
D4h.12	D4h.13	D4h.14	D4h.15	D4h.16	D4h.17	D4h.18
D4h.19	D4h.20	C3.1	C3.2	C3.3	C3.4	C3i.1
C3i.2	D3.1	D3.2	D3.3	D3.4	D3.5	D3.6
D3.7	C3v.1	C3v.2	C3v.3	C3v.4	C3v.5	C3v.6
D3d.1	D3d.2	D3d.3	D3d.4	D3d.5	D3d.6	C6.1
C6.2	C6.3	C6.4	C6.5	C6.6	C3h.1	C6h.1
C6h.2	D6.1	D6.2	D6.3	D6.4	D6.5	D6.6
C6v.1	C6v.2	C6v.3	C6v.4	D3h.1	D3h.2	D3h.3
D3h.4	D6h.1	D6h.2	D6h.3	D6h.4	T.1	T.2
T.3	T.4	T.5	Th.1	Th.2	Th.3	Th.4
Th.5	Th.6	Th.7	O.1	O.2	O.3	O.4
O.5	O.6	O.7	O.8	Td.1	Td.2	Td.3
Td.4	Td.5	Td.6	Oh.1	Oh.2	Oh.3	Oh.4
Oh.5	Oh.6	Oh.7	Oh.8	Oh.9	Oh.10	

Example: '`C2h.5`' (Schoenflies symbol for space group No. 14). **[space_group]**

_space_group.point_group_H-M (*char*)

The Hermann–Mauguin symbol denoting the geometric crystal class of space groups to which the space group belongs, and the geometric crystal class of point groups to which the point group of the space group belongs.

Examples: '`-4`', '`4/m`'. **[space_group]**

_space_group.reference_setting (*char*)

The reference setting of a given space group is the setting chosen by the International Union of Crystallography as a unique setting to which other settings can be referred using the transformation matrix column pair given in **_space_group.transform_Pp_abc** and **_space_group.transform_Qq_xyz**. The settings are given in the enumeration list in the form '**_space_group.IT_number: _space_group.name_Hall**'. The space-group number defines the space-group type and the Hall symbol specifies the symmetry generators referred to the reference coordinate system. The 230 reference settings chosen are identical to the settings listed in *International Tables for Crystallography* Volume A (2002). For the space groups where more than one setting is given in *International Tables*, the following choices have been made. For monoclinic space groups: unique axis *b* and cell choice 1. For space groups with two origins: origin choice 2 (origin at inversion centre, indicated by adding :2 to the Hermann–Mauguin symbol in

the enumeration list). For rhombohedral space groups: hexagonal axes (indicated by adding :h to the Hermann–Mauguin symbol in the enumeration list). Based on the symmetry table of R. W. Grosse-Kunstleve, ETH, Zurich. The enumeration list may be extracted from the dictionary and stored as a separate CIF that can be referred to as required.

In the enumeration list below, each reference setting is identified by Schoenflies symbol and by the Hermann–Mauguin symbol, augmented by :2 or :h suffixes as described above.

References: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers. Grosse-Kunstleve, R. W. (2001). *Xtal System of Crystallographic Programs, System Documentation*. http://xtal.crystal.uwa.edu.au/man/xtal3.7-228.html (or follow links to Docs→Space-Group Symbols from http://xtal.sourceforge.net).

The data value must be one of the following:

'001:P 1'	C_1^1	$P1$
'002:-P 1'	C_i^1	$P\bar{1}$
'003:P 2y'	C_2^1	$P121$
'004:P 2yb'	C_2^2	$P12_11$
'005:C 2y'	C_2^3	$C121$
'006:P -2y'	C_s^1	$P1m1$
'007:P -2yc'	C_s^2	$P1c1$
'008:C -2y'	C_s^3	$C1m1$
'009:C -2yc'	C_s^4	$C1c1$
'010:-P 2y'	C_{2h}^1	$P12/m1$
'011:-P 2yb'	C_{2h}^2	$P12_1/m1$
'012:-C 2y'	C_{2h}^3	$C12/m1$
'013:-P 2yc'	C_{2h}^4	$P12/c1$
'014:-P 2ybc'	C_{2h}^5	$P12_1/c1$
'015:-C 2yc'	C_{2h}^6	$C12/c1$
'016:P 2 2'	D_2^1	$P222$
'017:P 2c 2'	D_2^2	$P222_1$
'018:P 2 2ab'	D_2^3	$P2_12_12$
'019:P 2ac 2ab'	D_2^4	$P2_12_12_1$
'020:C 2c 2'	D_2^5	$C222_1$
'021:C 2 2'	D_2^6	$C222$
'022:F 2 2'	D_2^7	$F222$
'023:I 2 2'	D_2^8	$I222$
'024:I 2b 2c'	D_2^9	$I2_12_12_1$
'025:P 2 -2'	C_{2v}^1	$Pmm2$
'026:P 2c -2'	C_{2v}^2	$Pmc2_1$
'027:P 2 -2c'	C_{2v}^3	$Pcc2$
'028:P 2 -2a'	C_{2v}^4	$Pma2$
'029:P 2c -2ac'	C_{2v}^5	$Pca2_1$
'030:P 2 -2bc'	C_{2v}^6	$Pnc2$
'031:P 2ac -2'	C_{2v}^7	$Pmn2_1$
'032:P 2 -2ab'	C_{2v}^8	$Pba2$
'033:P 2c -2n'	C_{2v}^9	$Pna2_1$
'034:P 2 -2n'	C_{2v}^{10}	$Pnn2$
'035:C 2 -2'	C_{2v}^{11}	$Cmm2$
'036:C 2c -2'	C_{2v}^{12}	$Cmc2_1$
'037:C 2 -2c'	C_{2v}^{13}	$Ccc2$
'038:A 2 -2'	C_{2v}^{14}	$Amm2$
'039:A 2 -2b'	C_{2v}^{15}	$Aem2$
'040:A 2 -2a'	C_{2v}^{16}	$Ama2$
'041:A 2 -2ab'	C_{2v}^{17}	$Aea2$
'042:F 2 -2'	C_{2v}^{18}	$Fmm2$
'043:F 2 -2d'	C_{2v}^{19}	$Fdd2$
'044:I 2 -2'	C_{2v}^{20}	$Imm2$
'045:I 2 -2c'	C_{2v}^{21}	$Iba2$
'046:I 2 -2a'	C_{2v}^{22}	$Ima2$
'047:-P 2 2'	D_{2h}^1	$Pmmm$
'048:-P 2ab 2bc'	D_{2h}^2	$Pnnn$:2
'049:-P 2 2c'	D_{2h}^3	$Pccm$
'050:-P 2ab 2b'	D_{2h}^4	$Pban$:2
'051:-P 2a 2a'	D_{2h}^5	$Pmma$
'052:-P 2a 2bc'	D_{2h}^6	$Pnna$
'053:-P 2ac 2'	D_{2h}^7	$Pmna$
'054:-P 2a 2ac'	D_{2h}^8	$Pcca$
'055:-P 2 2ab'	D_{2h}^9	$Pbam$
'056:-P 2ab 2ac'	D_{2h}^{10}	$Pccn$
'057:-P 2c 2b'	D_{2h}^{11}	$Pbcm$
'058:-P 2 2n'	D_{2h}^{12}	$Pnnm$
'059:-P 2ab 2a'	D_{2h}^{13}	$Pmmn$:2
'060:-P 2n 2ab'	D_{2h}^{14}	$Pbcn$
'061:-P 2ac 2ab'	D_{2h}^{15}	$Pbca$
'062:-P 2ac 2n'	D_{2h}^{16}	$Pnma$
'063:-C 2c 2'	D_{2h}^{17}	$Cmcm$
'064:-C 2ac 2'	D_{2h}^{18}	$Cmce$
'065:-C 2 2'	D_{2h}^{19}	$Cmmm$
'066:-C 2 2c'	D_{2h}^{20}	$Cccm$
'067:-C 2a 2'	D_{2h}^{21}	$Cmme$
'068:-C 2a 2ac'	D_{2h}^{22}	$Ccce$:2
'069:-F 2 2'	D_{2h}^{23}	$Fmmm$
'070:-F 2uv 2vw'	D_{2h}^{24}	$Fddd$:2
'071:-I 2 2'	D_{2h}^{25}	$Immm$
'072:-I 2 2c'	D_{2h}^{26}	$Ibam$
'073:-I 2b 2c'	D_{2h}^{27}	$Ibca$
'074:-I 2b 2'	D_{2h}^{28}	$Imma$
'075:P 4'	C_4^1	$P4$
'076:P 4w'	C_4^2	$P4_1$
'077:P 4c'	C_4^3	$P4_2$
'078:P 4cw'	C_4^4	$P4_3$
'079:I 4'	C_4^5	$I4$
'080:I 4bw'	C_4^6	$I4_1$
'081:P -4'	S_4^1	$P\bar{4}$
'082:I -4'	S_4^2	$I\bar{4}$
'083:-P 4'	C_{4h}^1	$P4/m$
'084:-P 4c'	C_{4h}^2	$P4_2/m$
'085:-P 4a'	C_{4h}^3	$P4/n$:2
'086:-P 4bc'	C_{4h}^4	$P4_2/n$:2
'087:-I 4'	C_{4h}^5	$I4/m$
'088:-I 4ad'	C_{4h}^6	$I4_1/a$:2
'089:P 4 2'	D_4^1	$P422$
'090:P 4ab 2ab'	D_4^2	$P42_12$
'091:P 4w 2c'	D_4^3	$P4_122$
'092:P 4abw 2nw'	D_4^4	$P4_12_12$
'093:P 4c 2'	D_4^5	$P4_222$
'094:P 4n 2n'	D_4^6	$P4_22_12$
'095:P 4cw 2c'	D_4^7	$P4_322$
'096:P 4nw 2abw'	D_4^8	$P4_32_12$
'097:I 4 2'	D_4^9	$I422$
'098:I 4bw 2bw'	D_4^{10}	$I4_122$
'099:P 4 -2'	C_{4v}^1	$P4mm$
'100:P 4 -2ab'	C_{4v}^2	$P4bm$
'101:P 4c -2c'	C_{4v}^3	$P4_2cm$
'102:P 4n -2n'	C_{4v}^4	$P4_2nm$
'103:P 4 -2c'	C_{4v}^5	$P4cc$
'104:P 4 -2n'	C_{4v}^6	$P4nc$
'105:P 4c -2'	C_{4v}^7	$P4_2mc$

'106:P 4c -2ab'	C_{4v}^{8}	$P4_2bc$
'107:I 4 -2'	C_{4v}^{9}	$I4mm$
'108:I 4 -2c'	C_{4v}^{10}	$I4cm$
'109:I 4bw -2'	C_{4v}^{11}	$I4_1md$
'110:I 4bw -2c'	C_{4v}^{12}	$I4_1cd$
'111:P -4 2'	D_{2d}^{1}	$P\bar{4}2m$
'112:P -4 2c'	D_{2d}^{2}	$P\bar{4}2c$
'113:P -4 2ab'	D_{2d}^{3}	$P\bar{4}2_1m$
'114:P -4 2n'	D_{2d}^{4}	$P\bar{4}2_1c$
'115:P -4 -2'	D_{2d}^{5}	$P\bar{4}m2$
'116:P -4 -2c'	D_{2d}^{6}	$P\bar{4}c2$
'117:P -4 -2ab'	D_{2d}^{7}	$P\bar{4}b2$
'118:P -4 -2n'	D_{2d}^{8}	$P\bar{4}n2$
'119:I -4 -2'	D_{2d}^{9}	$I\bar{4}m2$
'120:I -4 -2c'	D_{2d}^{10}	$I\bar{4}c2$
'121:I -4 2'	D_{2d}^{11}	$I\bar{4}2m$
'122:I -4 2bw'	D_{2d}^{12}	$I\bar{4}2d$
'123:-P 4 2'	D_{4h}^{1}	$P4/mmm$
'124:-P 4 2c'	D_{4h}^{2}	$P4/mcc$
'125:-P 4a 2b'	D_{4h}^{3}	$P4/nbm$:2
'126:-P 4a 2bc'	D_{4h}^{4}	$P4/nnc$:2
'127:-P 4 2ab'	D_{4h}^{5}	$P4/mbm$
'128:-P 4 2n'	D_{4h}^{6}	$P4/mnc$
'129:-P 4a 2a'	D_{4h}^{7}	$P4/nmm$:2
'130:-P 4a 2ac'	D_{4h}^{8}	$P4/ncc$:2
'131:-P 4c 2'	D_{4h}^{9}	$P4_2/mmc$
'132:-P 4c 2c'	D_{4h}^{10}	$P4_2/mcm$
'133:-P 4ac 2b'	D_{4h}^{11}	$P4_2/nbc$:2
'134:-P 4ac 2bc'	D_{4h}^{12}	$P4_2/nnm$:2
'135:-P 4c 2ab'	D_{4h}^{13}	$P4_2/mbc$
'136:-P 4n 2n'	D_{4h}^{14}	$P4_2/mnm$
'137:-P 4ac 2a'	D_{4h}^{15}	$P4_2/nmc$:2
'138:-P 4ac 2ac'	D_{4h}^{16}	$P4_2/ncm$:2
'139:-I 4 2'	D_{4h}^{17}	$I4/mmm$
'140:-I 4 2c'	D_{4h}^{18}	$I4/mcm$
'141:-I 4bd 2'	D_{4h}^{19}	$I4_1/amd$:2
'142:-I 4bd 2c'	D_{4h}^{20}	$I4_1/acd$:2
'143:P 3'	C_{3}^{1}	$P3$
'144:P 31'	C_{3}^{2}	$P3_1$
'145:P 32'	C_{3}^{3}	$P3_2$
'146:R 3'	C_{3}^{4}	$R3$:h
'147:-P 3'	C_{3i}^{1}	$P\bar{3}$
'148:-R 3'	C_{3i}^{2}	$R\bar{3}$:h
'149:P 3 2'	D_{3}^{1}	$P312$
'150:P 3 2"'	D_{3}^{2}	$P321$
'151:P 31 2 (0 0 4)'	D_{3}^{3}	$P3_112$
'152:P 31 2"'	D_{3}^{4}	$P3_121$
'153:P 32 2 (0 0 2)'	D_{3}^{5}	$P3_212$
'154:P 32 2"'	D_{3}^{6}	$P3_221$
'155:R 3 2"'	D_{3}^{7}	$R32$:h
'156:P 3 -2"'	C_{3v}^{1}	$P3m1$
'157:P 3 -2'	C_{3v}^{2}	$P31m$
'158:P 3 -2"c'	C_{3v}^{3}	$P3c1$
'159:P 3 -2c'	C_{3v}^{4}	$P31c$
'160:R 3 -2"'	C_{3v}^{5}	$R3m$:h
'161:R 3 -2"c'	C_{3v}^{6}	$R3c$:h
'162:-P 3 2'	D_{3d}^{1}	$P\bar{3}1m$
'163:-P 3 2c'	D_{3d}^{2}	$P\bar{3}1c$
'164:-P 3 2"'	D_{3d}^{3}	$P\bar{3}m1$
'165:-P 3 2"c'	D_{3d}^{4}	$P\bar{3}c1$
'166:-R 3 2"'	D_{3d}^{5}	$R\bar{3}m$:h
'167:-R 3 2"c'	D_{3d}^{6}	$R\bar{3}c$:h
'168:P 6'	C_{6}^{1}	$P6$
'169:P 61'	C_{6}^{2}	$P6_1$
'170:P 65'	C_{6}^{3}	$P6_5$
'171:P 62'	C_{6}^{4}	$P6_2$
'172:P 64'	C_{6}^{5}	$P6_4$
'173:P 6c'	C_{6}^{6}	$P6_3$
'174:P -6'	C_{3h}^{1}	$P\bar{6}$
'175:-P 6'	C_{6h}^{1}	$P6/m$
'176:-P 6c'	C_{6h}^{2}	$P6_3/m$
'177:P 6 2'	D_{6}^{1}	$P622$
'178:P 61 2 (0 0 5)'	D_{6}^{2}	$P6_122$
'179:P 65 2 (0 0 1)'	D_{6}^{3}	$P6_522$
'180:P 62 2 (0 0 4)'	D_{6}^{4}	$P6_222$
'181:P 64 2 (0 0 2)'	D_{6}^{5}	$P6_422$
'182:P 6c 2c'	D_{6}^{6}	$P6_322$
'183:P 6 -2'	C_{6v}^{1}	$P6mm$
'184:P 6 -2c'	C_{6v}^{2}	$P6cc$
'185:P 6c -2'	C_{6v}^{3}	$P6_3cm$
'186:P 6c -2c'	C_{6v}^{4}	$P6_3mc$
'187:P -6 2'	D_{3h}^{1}	$P\bar{6}m2$
'188:P -6c 2'	D_{3h}^{2}	$P\bar{6}c2$
'189:P -6 -2'	D_{3h}^{3}	$P\bar{6}2m$
'190:P -6c -2c'	D_{3h}^{4}	$P\bar{6}2c$
'191:-P 6 2'	D_{6h}^{1}	$P6/mmm$
'192:-P 6 2c'	D_{6h}^{2}	$P6/mcc$
'193:-P 6c 2'	D_{6h}^{3}	$P6_3/mcm$
'194:-P 6c 2c'	D_{6h}^{4}	$P6_3/mmc$
'195:P 2 2 3'	T^{1}	$P23$
'196:F 2 2 3'	T^{2}	$F23$
'197:I 2 2 3'	T^{3}	$I23$
'198:P 2ac 2ab 3'	T^{4}	$P2_13$
'199:I 2b 2c 3'	T^{5}	$I2_13$
'200:-P 2 2 3'	T_{h}^{1}	$Pm\bar{3}$
'201:-P 2ab 2bc 3'	T_{h}^{2}	$Pn\bar{3}$:2
'202:-F 2 2 3'	T_{h}^{3}	$Fm\bar{3}$
'203:-F 2uv 2vw 3'	T_{h}^{4}	$Fd\bar{3}$:2
'204:-I 2 2 3'	T_{h}^{5}	$Im\bar{3}$
'205:-P 2ac 2ab 3'	T_{h}^{6}	$Pa\bar{3}$
'206:-I 2b 2c 3'	T_{h}^{7}	$Ia\bar{3}$
'207:P 4 2 3'	O^{1}	$P432$
'208:P 4n 2 3'	O^{2}	$P4_232$
'209:F 4 2 3'	O^{3}	$F432$
'210:F 4d 2 3'	O^{4}	$F4_132$
'211:I 4 2 3'	O^{5}	$I432$
'212:P 4acd 2ab 3'	O^{6}	$P4_332$
'213:P 4bd 2ab 3'	O^{7}	$P4_132$
'214:I 4bd 2c 3'	O^{8}	$I4_132$
'215:P -4 2 3'	T_{d}^{1}	$P\bar{4}3m$
'216:F -4 2 3'	T_{d}^{2}	$F\bar{4}3m$
'217:I -4 2 3'	T_{d}^{3}	$I\bar{4}3m$
'218:P -4n 2 3'	T_{d}^{4}	$P\bar{4}3n$
'219:F -4a 2 3'	T_{d}^{5}	$F\bar{4}3c$
'220:I -4bd 2c 3'	T_{d}^{6}	$I\bar{4}3d$
'221:-P 4 2 3'	O_{h}^{1}	$Pm\bar{3}m$

'222:-P 4a 2bc 3'	O_h^2	$P\,n\,\bar{3}\,n$:2
'223:-P 4n 2 3'	O_h^3	$P\,m\,\bar{3}\,n$
'224:-P 4bc 2bc 3'	O_h^4	$P\,n\,\bar{3}\,m$:2
'225:-F 4 2 3'	O_h^5	$F\,m\,\bar{3}\,m$
'226:-F 4a 2 3'	O_h^6	$F\,m\,\bar{3}\,c$
'227:-F 4vw 2vw 3'	O_h^7	$F\,d\,\bar{3}\,m$:2
'228:-F 4ud 2vw 3'	O_h^8	$F\,d\,\bar{3}\,c$:2
'229:-I 4 2 3'	O_h^9	$I\,m\,\bar{3}\,m$
'230:-I 4bd 2c 3'	O_h^{10}	$I\,a\,\bar{3}\,d$

[space_group]

_space_group.transform_Pp_abc

This item specifies the transformation $(P,\mathbf{p})$ of the basis vectors from the setting used in the CIF (a,b,c) to the reference setting given in **_space_group.reference_setting** (a',b',c'). The value is given in Jones–Faithful notation corresponding to the rotational matrix P combined with the origin shift vector $\mathbf{p}$ in the expression

$$(a',b',c') = (a,b,c)P + \mathbf{p}.$$

P is a post-multiplication matrix of a row (a,b,c) of column vectors. It is related to the inverse transformation $(Q,\mathbf{q})$ by

$$P = Q^{-1},$$
$$\mathbf{p} = Pq = -(Q^{-1})\mathbf{q}.$$

These transformations are applied as follows: atomic coordinates $(x',y',z') = Q(x,y,z) + \mathbf{q}$, Miller indices $(h',k',l') = (h,k,l)P$, symmetry operations $W' = (Q,\mathbf{q})W(P,\mathbf{p})$, basis vectors $(a',b',c') = (a,b,c)P + \mathbf{p}$.

This item is given as a character string involving the characters a, b and c with commas separating the expressions for the a', b' and c' vectors. The numeric values may be given as integers, fractions or real numbers. Multiplication is implicit, division must be explicit. White space within the string is optional.

Examples: '`-b+c, a+c, -a+b+c`' (*R*3:r to *R*3:h), '`a-1/4, b-1/4, c-1/4`' (*Pnnn*:1 to *Pnnn*:2), '`b-1/2, c-1/2, a-1/2`' (*Bbab*:1 to *Ccca*:2). **[space_group]**

_space_group.transform_Qq_xyz

This item specifies the transformation $(Q,\mathbf{q})$ of the atomic coordinates from the setting used in the CIF [(x,y,z) referred to the basis vectors (a,b,c)] to the reference setting given in **_space_group.reference_setting** [(x',y',z') referred to the basis vectors (a',b',c')]. The value given in Jones–Faithful notation corresponds to the rotational matrix Q combined with the origin shift vector $\mathbf{q}$ in the expression

$$(x',y',z') = Q(x,y,z) + \mathbf{q}.$$

Q is a pre-multiplication matrix of the column vector (x,y,z). It is related to the inverse transformation $(P,\mathbf{p})$ by

$$P = Q^{-1},$$
$$\mathbf{p} = P\mathbf{q} = -(Q^{-1})\mathbf{q},$$

where the P and Q transformations are applied as follows: atomic coordinates $(x',y',z') = Q(x,y,z) + \mathbf{q}$, Miller indices $(h',k',l') = (h,k,l)P$, symmetry operations $W' = (Q,\mathbf{q})W(P,\mathbf{p})$, basis vectors $(a',b',c') = (a,b,c)P + \mathbf{p}$.

This item is given as a character string involving the characters x, y and z with commas separating the expressions for the x', y' and z' components. The numeric values may be given as integers, fractions or real numbers. Multiplication is implicit, division must be explicit. White space within the string is optional.

Examples: '`-x/3+2y/3-z/3, -2x/3+y/3+z/3, x/3+y/3+z/3`' (*R*3:r to *R*3:h), '`x+1/4,y+1/4,z+1/4`' (*Pnnn*:1 to *Pnnn*:2), '`z+1/2,x+1/2,y+1/2`' (*Bbab*:1 to *Ccca*:2). **[space_group]**

SPACE_GROUP_SYMOP

Contains information about the symmetry operations of the space group.

Category key(s): **_space_group_symop.id**

Example 1 – the symmetry operations for the space group $P2_1/c$.

```
loop_
_space_group_symop.id
_space_group_symop.operation_xyz
_space_group_symop.operation_description
  1    x,y,z              'identity mapping'
  2    -x,-y,-z           'inversion'
  3    -x,1/2+y,1/2-z
         '2-fold screw rotation with axis in (0,y,1/4)'
  4    x,1/2-y,1/2+z
         'c glide reflection through the plane (x,1/4,y)'
```

_space_group_symop.generator_xyz (*char*)

A parsable string giving one of the symmetry generators of the space group in algebraic form. If W is a matrix representation of the rotational part of the generator defined by the positions and signs of x, y and z, and $\mathbf{w}$ is a column of translations defined by the fractions, an equivalent position $\boldsymbol{x}'$ is generated from a given position $\boldsymbol{x}$ by

$$\boldsymbol{x}' = W\boldsymbol{x} + \mathbf{w}.$$

When a list of symmetry generators is given, it is assumed that the complete list of symmetry operations of the space group (including the identity operation) can be generated through repeated multiplication of the generators, that is, $(W_3,\mathbf{w}_3)$ is an operation of the space group if $(W_2,\mathbf{w}_2)$ and $(W_1,\mathbf{w}_1)$ [where $(W_1,\mathbf{w}_1)$ is applied first] are either operations or generators and

$$W_3 = W_2 \times W_1,$$
$$\mathbf{w}_3 = W_2 \times \mathbf{w}_1 + \mathbf{w}_2.$$

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Related item: **_space_group_symop.operation_xyz** (alternate). Where no value is given, the assumed value is '`x,y,z`'.

Example: '`x,1/2-y,1/2+z`' (*c* glide reflection through the plane (x, 1/4, z) chosen as one of the generators of the space group). **[space_group_symop]**

***_space_group_symop.id** (*char*)

_symmetry_equiv_pos_site_id (*cif_core.dic 1.0*)

_symmetry_equiv.id (*cif_mm.dic 1.0*)

An arbitrary identifier that uniquely labels each symmetry operation in the list.

[space_group_symop]

_space_group_symop.operation_description (*char*)

An optional text description of a particular symmetry operation of the space group.

[space_group_symop]

_space_group_symop.operation_xyz (*char*)

_symmetry_equiv_pos_as_xyz (*cif_core.dic 1.0*)

A parsable string giving one of the symmetry operations of the space group in algebraic form. If W is a matrix representation of the rotational part of the symmetry operation defined by the positions and signs of x, y and z, and $\mathbf{w}$ is a column of translations defined by the fractions, an equivalent position $\boldsymbol{x}'$ is generated from a given position $\boldsymbol{x}$ by

$$\boldsymbol{x}' = W\boldsymbol{x} + \mathbf{w}.$$

When a list of symmetry operations is given, it is assumed that the list contains all the operations of the space group (including the identity operation) as given by the representatives of the general position in *International Tables for Crystallography* Volume A.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Related item: **_space_group_symop.generator_xyz** (alternate). Where no value is given, the assumed value is 'x,y,z'.

Example: 'x,1/2-y,1/2+z' (*c* glide reflection through the plane (*x*, 1/4, *z*)).

[space_group_symop]

_space_group_symop.sg_id (*numb*)

A child of **_space_group.id** allowing the symmetry operation to be identified with a particular space group.

[space_group_symop]

SPACE_GROUP_WYCKOFF

Contains information about Wyckoff positions of a space group. Only one site can be given for each special position but the remainder can be generated by applying the symmetry operations stored in **_space_group_symop.operation_xyz.**

Category key(s): **_space_group_Wyckoff.id**

Example 1 – this example is taken from the space group $Fd\bar{3}c$ (No. 228, origin choice 2). For brevity only a selection of special positions are listed. The coordinates of only one site per special position can be given in this item, but the coordinates of the other sites can be generated using the symmetry operations given in the SPACE_GROUP_SYMOP category.

```
loop_
_space_group_Wyckoff.id
_space_group_Wyckoff.multiplicity
_space_group_Wyckoff.letter
_space_group_Wyckoff.site_symmetry
_space_group_Wyckoff.coord_xyz
    1  192   h   1      x,y,z
    2   96   g   ..2    1/4,y,-y
    3   96   f   2..    x,1/8,1/8
    4   32   b   .32    1/4,1/4,1/4
```

_space_group_Wyckoff.coords_xyz (*char*)

Coordinates of one site of a Wyckoff position expressed in terms of its fractional coordinates (*x*, *y*, *z*) in the unit cell. To generate the coordinates of all sites of this Wyckoff position, it is necessary to multiply these coordinates by the symmetry operations stored in **_space_group_symop.operation_xyz.**

Where no value is given, the assumed value is 'x,y,z'.

Example: 'x,1/2,0' (coordinates of Wyckoff site with 2.. symmetry).

[space_group_Wyckoff]

* **_space_group_Wyckoff.id** (*char*)

An arbitrary identifier that is unique to a particular Wyckoff position.

[space_group_Wyckoff]

_space_group_Wyckoff.letter (*char*)

The Wyckoff letter associated with this position, as given in *International Tables for Crystallography* Volume A. The enumeration value \a corresponds to the Greek letter 'α' used in *International Tables*.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

The data value must be one of the following:

a b c d e f g h i j k l m n o p q r s t u v w x y z \a

[space_group_Wyckoff]

_space_group_Wyckoff.multiplicity (*numb*)

The multiplicity of this Wyckoff position as given in *International Tables* Volume A. It is the number of equivalent sites per conventional unit cell.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

The permitted range is $[1, \infty)$. **[space_group_Wyckoff]**

_space_group_Wyckoff.sg_id (*char*)

A child of **_space_group.id** allowing the Wyckoff position to be identified with a particular space group.

[space_group_Wyckoff]

_space_group_Wyckoff.site_symmetry (*char*)

The subgroup of the space group that leaves the point fixed. It is isomorphic to a subgroup of the point group of the space group. The site-symmetry symbol indicates the symmetry in the symmetry direction determined by the Hermann–Mauguin symbol of the space group (see *International Tables for Crystallography* Volume A, Section 2.2.12).

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Examples: '2.22' (position 2*b* in space group No. 94, $P4_22_12$), '42.2' (position 6*b* in space group No. 222, $Pn\bar{3}n$), '2..' (Site symmetry for the Wyckoff position 96*f* in space group No. 228, $Fd\bar{3}c$. The site-symmetry group is isomorphic to the point group 2 with the twofold axis along one of the 100 directions.). **[space_group_Wyckoff]**

4.8. Molecular Information File dictionary (MIF)

BY F. H. ALLEN, J. M. BARNARD, A. P. F. COOK AND S. R. HALL

This is version 1.2.3 of the molecular information file dictionary (MIF). MIF complements the crystallographic information file with data items describing two-dimensional chemical structure diagrams and bond types in a similar STAR format. Unlike CIF, MIF permits nested looping of data items and data encapsulation in save frames. The MIF format is described in Chapter 2.4.

_atom_attach_all
_atom_attach_ring
_atom_attach_nh
_atom_attach_h (*numb*)

The number of atom sites considered to be attached (*i.e.* chemically bonded) to this site. The extensions are:

all	all sites,
ring	all sites forming rings,
nh	all sites excluding hydrogens and unshared electron pairs,
h	hydrogen-atom sites.

Appears in list containing **_atom_id**.
The permitted range is $0 \rightarrow \infty$. **[atom]**

_atom_charge (*numb*)

Specifies the formal electronic charge on the atom for the different atomic representation conventions. The convention for charge is specified by **_define_bonding_convention**.
Appears in list containing **_atom_id**.
The permitted range is $-99 \rightarrow 99$. **[atom]**

_atom_cip (*char*)

Specifies the Cahn–Ingold–Prelog designation for the atom. The designators are by Prelog & Helmchen (1982).

Reference: Prelog, V. & Helmchen, G. (1982). *Angew. Chem. Int. Ed. Engl.* **21**, 567–583.

Appears in list containing **_atom_id**. **[atom]**

_atom_coord_x
_atom_coord_y
_atom_coord_z (*numb*)

Specifies the Cartesian coordinates for the atom with an arbitrary origin and arbitrary orthogonal axes.
Appears in list containing **_atom_id**. **[atom]**

_atom_environment (*char*)

Environment of the atom site.
Appears in list containing **_atom_id**.
The data value must be one of the following:

atom	a single atom site
frag	atom site in molecular fragment

Where no value is given, the assumed value is 'atom'. **[atom]**

_atom_frag_id (*numb*)

ID of atom site within the molecular fragment.
Appears in list containing **_atom_id**. **[atom]**

_atom_frag_key (*char*)

Name of save frame containing atom data on molecular fragment.
Appears in list containing **_atom_id**. **[atom]**

_atom_id (*char*)

This specifies a unique code for an 'atom site' in a molecule or fragment. A designated atom site may be occupied by a 'dummy' atom (see **_atom_type**). A special syntax exists for this code which permits a template molecular fragment to be referred to as an 'atom site', and for the atom within that template to be identified. The syntax is $m > n$ where m is the code identifying the template fragment (contained within a save frame) and n is the code matching an **_atom_id** value stored within the save frame.
Appears in list as essential element of loop structure. May match child data name(s):
_bond_id_1, **_bond_id_2**, **_stereo_vertex_id**. **[atom]**

_atom_label (*char*)

The code providing unique identification of the atom site for display purposes. See also **_atom_type**.
Appears in list containing **_atom_id**. **[atom]**

_atom_mass_number (*numb*)

Specifies the mass number or the isotopic state of the atom. The default is the 'most abundant naturally occurring' value.
Appears in list containing **_atom_id**.
The permitted range is $1 \rightarrow \infty$. **[atom]**

_atom_radical_count (*numb*)

Specifies the formal radical state of the atom site identified by **_atom_id**. This is the number of unpaired electrons associated with the atom, *e.g.* the value of 1 specifies one singly occupied orbital on the atom that is not involved in bonding.
Appears in list containing **_atom_id**.
The permitted range is $0 \rightarrow \infty$. **[atom]**

_atom_spin_multiplicity (*numb*)

The spin multiplicity applies only if the **_atom_radical_count** value is nonzero, and is derived from the formula $2S + 1$, where S is the absolute value of the sum of the electron spins ($+\frac{1}{2}$ or $-\frac{1}{2}$) on the atom. Thus, a biradical atom may have the values 1 or 3 [$2(\frac{1}{2} - \frac{1}{2}) + 1$ or $2(\frac{1}{2} + \frac{1}{2}) + 1$] to represent a singlet or a triplet biradical.
Appears in list containing **_atom_id**.
The permitted range is $1 \rightarrow \infty$. **[atom]**

_atom_type (*char*)

Specifies the type of atom site identified by **_atom_id**. Hydrogen atoms will usually not be assigned to sites except where they are connected to more than one other atom, or where it is necessary to reference them, for example, in describing stereochemistry.
The data value must be one of the following:

H	1	He	2	Li	3	Be	4	B	5	C	6
N	7	O	8	F	9	Ne	10	Na	11	Mg	12
Al	13	Si	14	P	15	S	16	Cl	17	Ar	18
K	19	Ca	20	Sc	21	Ti	22	V	23	Cr	24
Mn	25	Fe	26	Co	27	Ni	28	Cu	29	Zn	30
Ga	31	Ge	32	As	33	Se	34	Br	35	Kr	36
Rb	37	Sr	38	Y	39	Zr	40	Nb	41	Mo	42
Tc	43	Ru	44	Rh	45	Pd	46	Ag	47	Cd	48
In	49	Sn	50	Sb	51	Te	52	I	53	Xe	54
Cs	55	Ba	56	La	57	Ce	58	Pr	59	Nd	60
Pm	61	Sm	62	Eu	63	Gd	64	Tb	65	Dy	66
Ho	67	Er	68	Tm	69	Yb	70	Lu	71	Hf	72
Ta	73	W	74	Re	75	Os	76	Ir	77	Pt	78
Au	79	Hg	80	Tl	81	Pb	82	Bi	83	Po	84
At	85	Rn	86	Fr	87	Ra	88	Ac	89	Th	90

Affiliations: FRANK H. ALLEN, Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge, CB2 1EZ, England; JOHN M. BARNARD and ANTHONY P. F. COOK, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia.

Pa	91	U	92	Np	93	Pu	94	Am	95	Cm	96
Bk	97	Cf	98	Es	99	Fm	100	Md	101	No	102
Lr	103	usp	'unshared electron pair'					dum	'dummy atom'		

Appears in list containing **_atom_id**. **[atom]**

_atom_valency (*numb*)

Specifies the valency state of the atom site identified by **_atom_id**.

Appears in list containing **_atom_id**.

The permitted range is $-99 \rightarrow 99$. **[atom]**

_bond_cip (*char*)

Specifies the Cahn–Ingold–Prelog designation for the bond. The designators are by Prelog & Helmchen (1982).

Reference: Prelog, V. & Helmchen, G. (1982). *Angew. Chem. Int. Ed. Engl.* **21**, 567–583.

Appears in list containing **_bond_id_1**, **_bond_id_2**. **[bond]**

_bond_environment (*char*)

Specifies the connection environment of the bond.

Appears in list containing **_bond_id_1**, **_bond_id_2**.

The data value must be one of the following:

ring	in a ring
chain	in a chain

[bond]

_bond_id_1
_bond_id_2 (*char*)

Specify the atom-site codes of 'chemically connected' sites. The atom ID codes may appear in either order except for asymmetric bonds. Each 'bond' pair may be represented only once. The special syntax for representing atom sites within molecular templates is described in the **_atom_id** definition.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_id**. **[bond]**

_bond_type_casreg3 (*char*)

Code indicating the nature of the bond according to the *Chemical Abstracts* Service Registry III bonding convention. The convention is specified with **_define_bonding_convention**.

Reference: Mockus, J. & Stobaugh, R. E. (1980). *J. Chem. Inf. Comput. Sci.* **20**, 18–22.

Appears in list containing **_bond_id_1**, **_bond_id_2**.

Related items:

_bond_type_mif (convention),
_bond_type_ccdc (convention).

The data value must be one of the following:

S	single exact bond
D	double exact bond
T	triple exact bond
A	ring alternating normalized bond
U	tautomer normalized bond

[bond]

_bond_type_ccdc (*char*)

Code indicating the nature of the bond according to the Cambridge Crystallographic Data Centre (CCDC) bonding convention. The convention is specified with **_define_bonding_convention**.

Reference: Allen, F. H., Davies, J. E., Galloy, J. J., Johnson, O., Kennard, O., Macrae, C. F., Mitchell, E. M., Mitchell, G. F., Smith, J. M. & Watson, D. G. (1991). *J. Chem. Inf. Comput. Sci.* **31**, 187–204.

Appears in list containing **_bond_id_1**, **_bond_id_2**.

Related items:

_bond_type_casreg3 (convention),
_bond_type_mif (convention).

The data value must be one of the following:

S	single (two-electron) bond or σ bond to metal
D	double (four-electron) bond
T	triple (six-electron) bond
Q	quadruple (eight-electron, metal–metal) bond
A	alternating normalized ring bond (aromatic)
C	catena-forming bond in crystal structure
E	equivalent (delocalized double) bond
P	π bond (metal–ligand π interaction)

[bond]

_bond_type_mif (*char*)

Code indicating the nature of the bond according to the MIF bonding convention. Aromatic and normalized tautomeric bonds cannot be shown. The convention is specified with **_define_bonding_convention**.

Appears in list containing **_bond_id_1**, **_bond_id_2**.

Related items:

_bond_type_casreg3 (convention),
_bond_type_ccdc (convention).

The data value must be one of the following:

S	single (two-electron) bond
D	double (four-electron) bond
T	triple (six-electron) bond
O	other (*e.g.* coordination) bond

[bond]

_define_bonding_convention (*char*)

Specifies the convention used for the values of the items **_atom_charge**, **_atom_radical_count**, **_atom_spin_multiplicity** and **_atom_valency**.

The data value must be one of the following:

casreg3	*Chemical Abstracts* Service Registry III
ccdc	Cambridge Crystallographic Data Centre
mif	basic MIF

Where no value is given, the assumed value is 'mif'. **[define]**

_define_stereo_relationship (*char*)

Defines the enantiomorphic relationship of the stereo geometry in the data cell (data block or save frame). The descriptions of independently defined stereo regions, whose centres all have the same inter-centre relationship, may be grouped together. For each such group, the inter-centre relationship is specified by **_define_stereo_relationship**. Where there is only one such group, all the relevant data may be included in the data block or save frame for the molecule as a whole. Where there are several different such groups, each should be shown in a separate save frame and **_reference_stereo_group** should be used to reference the different save frames. For each stereo group, a two-level loop structure will be used to define the stereocentres it contains. Each stereogenic atom site is defined by **_stereo_atom_id** and **_stereo_geometry** at the first loop level, and by **_stereo_vertex_id** at the second. Each stereogenic bond is defined by **_stereo_bond_id_1**, **_stereo_bond_id_2** and **_stereo_geometry** at the first level, and by **_stereo_vertex_id** at the second level.

The data value must be one of the following:

absolute	configuration is as shown
relative	configuration is relative
unknown	configuration is unknown
racemic	two stereoisomers: equal mix of D and L
absolute_excess	two stereoisomers; excess as shown
relative_excess	two stereoisomers; excess unknown

Where no value is given, the assumed value is 'unknown'. **[define]**

_display_colour (*char*)

The colour code specifying the colour of the object identified by **_display_symbol**. The permitted colour codes are stored with the RGB ratios as a separate validation file 'mif_core_colours.val'.

The data value must be one of the following:

black	000:000:000_RGB	white	255:255:255_RGB
white_snow	255:250:250_RGB	white_smoke	245:245:245_RGB
white_ivory	255:255:240_RGB	white_azure	240:255:255_RGB
white_lavender	230:230:250_RGB	white_rose	255:228:225_RGB
grey	192:192:192_RGB	grey_light	211:211:211_RGB

grey_slate	112:128:144_RGB	grey_slate_dark	047:079:079_RGB
grey_slate_light	119:136:153_RGB	blue	000:000:255_RGB
blue_light	176:224:230_RGB	blue_medium	000:000:205_RGB
blue_dark	025:025:112_RGB	blue_navy	000:000:128_RGB
blue_slate_dark	072:061:139_RGB	blue_slate	106:090:205_RGB
blue_royal	065:105:225_RGB	blue_sky	135:206:235_RGB
blue_sky_deep	000:191:255_RGB	blue_sky_light	135:206:250_RGB
blue_steel	070:130:180_RGB	turquoise	064:224:208_RGB
turquoise_light	175:238:238_RGB	turquoise_dark	000:206:209_RGB
turquoise_medium	072:209:204_RGB	cyan	000:255:255_RGB
cyan_light	224:255:255_RGB	green	000:255:000_RGB
green_light	152:251:152_RGB	green_dark	000:100:000_RGB
green_sea	046:139:087_RGB	green_sea_dark	143:188:143_RGB
green_sea_medium	060:179:113_RGB	green_sea_light	032:178:170_RGB
green_spring	000:255:127_RGB	green_lawn	124:252:000_RGB
green_aquamarine	127:255:212_RGB	green_chartreuse	127:255:000_RGB
green_yellow	173:255:047_RGB	green_lime	050:205:050_RGB
green_forest	034:139:034_RGB	green_olive	107:142:035_RGB
green_khaki	240:230:140_RGB	yellow	255:255:000_RGB
yellow_light	255:255:224_RGB	yellow_gold	255:215:000_RGB
yellow_goldenrod	218:165:032_RGB	yellow_goldenrod_pale	238:232:170_RGB
yellow_goldenrod_light	238:221:130_RGB	yellow_goldenrod_dark	184:134:011_RGB
yellow_green	154:205:050_RGB	brown	165:042:042_RGB
brown_rosy	188:143:143_RGB	brown_indian_red	205:092:092_RGB
brown_saddle	139:069:019_RGB	brown_sienna	160:082:045_RGB
brown_peru	205:133:063_RGB	brown_burlywood	222:184:135_RGB
brown_beige	245:245:220_RGB	brown_wheat	245:222:179_RGB
brown_sandy	244:164:096_RGB	brown_tan	210:180:140_RGB
brown_chocolate	210:105:030_RGB	salmon	250:128:114_RGB
salmon_light	255:160:122_RGB	salmon_dark	233:150:122_RGB
orange	255:165:000_RGB	orange_dark	255:140:000_RGB
red	255:000:000_RGB	red_coral	255:127:080_RGB
red_tomato	255:099:071_RGB	red_orange	255:069:000_RGB
red_violet	219:112:147_RGB	red_maroon	176:048:096_RGB
pink	255:192:203_RGB	pink_light	255:182:193_RGB
pink_deep	255:020:147_RGB	pink_hot	255:105:180_RGB
violet	238:130:238_RGB	violet_red_medium	199:021:133_RGB
violet_red	208:032:144_RGB	violet_magenta	255:000:255_RGB
violet_plum	221:160:221_RGB	violet_orchid	218:112:214_RGB
violet_dark	148:000:211_RGB	violet_blue	138:043:226_RGB
violet_purple	160:032:240_RGB	violet_thistle	216:191:216_RGB

Appears in list containing **_display_id**. Where no value is given, the assumed value is 'black'. **[display]**

_display_conn_colour (*char*)

The colour code specifying the colour of the object identified by **_display_conn_symbol**. The permitted colour codes are stored with the RGB ratios as a separate validation file 'mif_core_colours.val'.

The data value must be one of the following:

black	000:000:000_RGB	white	255:255:255_RGB
white_snow	255:250:250_RGB	white_smoke	245:245:245_RGB
white_ivory	255:255:240_RGB	white_azure	240:255:255_RGB
white_lavender	230:230:250_RGB	white_rose	255:228:225_RGB
grey	192:192:192_RGB	grey_light	211:211:211_RGB
grey_slate	112:128:144_RGB	grey_slate_dark	047:079:079_RGB
grey_slate_light	119:136:153_RGB	blue	000:000:255_RGB
blue_light	176:224:230_RGB	blue_medium	000:000:205_RGB
blue_dark	025:025:112_RGB	blue_navy	000:000:128_RGB
blue_slate_dark	072:061:139_RGB	blue_slate	106:090:205_RGB
blue_royal	065:105:225_RGB	blue_sky	135:206:235_RGB
blue_sky_deep	000:191:255_RGB	blue_sky_light	135:206:250_RGB
blue_steel	070:130:180_RGB	turquoise	064:224:208_RGB
turquoise_light	175:238:238_RGB	turquoise_dark	000:206:209_RGB
turquoise_medium	072:209:204_RGB	cyan	000:255:255_RGB
cyan_light	224:255:255_RGB	green	000:255:000_RGB
green_light	152:251:152_RGB	green_dark	000:100:000_RGB
green_sea	046:139:087_RGB	green_sea_dark	143:188:143_RGB
green_sea_medium	060:179:113_RGB	green_sea_light	032:178:170_RGB
green_spring	000:255:127_RGB	green_lawn	124:252:000_RGB
green_aquamarine	127:255:212_RGB	green_chartreuse	127:255:000_RGB
green_yellow	173:255:047_RGB	green_lime	050:205:050_RGB
green_forest	034:139:034_RGB	green_olive	107:142:035_RGB
green_khaki	240:230:140_RGB	yellow	255:255:000_RGB
yellow_light	255:255:224_RGB	yellow_gold	255:215:000_RGB
yellow_goldenrod	218:165:032_RGB	yellow_goldenrod_pale	238:232:170_RGB
yellow_goldenrod_light	238:221:130_RGB	yellow_goldenrod_dark	184:134:011_RGB
yellow_green	154:205:050_RGB	brown	165:042:042_RGB
brown_rosy	188:143:143_RGB	brown_indian_red	205:092:092_RGB
brown_saddle	139:069:019_RGB	brown_sienna	160:082:045_RGB
brown_peru	205:133:063_RGB	brown_burlywood	222:184:135_RGB
brown_beige	245:245:220_RGB	brown_wheat	245:222:179_RGB
brown_sandy	244:164:096_RGB	brown_tan	210:180:140_RGB
brown_chocolate	210:105:030_RGB	salmon	250:128:114_RGB
salmon_light	255:160:122_RGB	salmon_dark	233:150:122_RGB
orange	255:165:000_RGB	orange_dark	255:140:000_RGB
red	255:000:000_RGB	red_coral	255:127:080_RGB
red_tomato	255:099:071_RGB	red_orange	255:069:000_RGB
red_violet	219:112:147_RGB	red_maroon	176:048:096_RGB
pink	255:192:203_RGB	pink_light	255:182:193_RGB
pink_deep	255:020:147_RGB	pink_hot	255:105:180_RGB
violet	238:130:238_RGB	violet_red_medium	199:021:133_RGB
violet_red	208:032:144_RGB	violet_magenta	255:000:255_RGB
violet_plum	221:160:221_RGB	violet_orchid	218:112:214_RGB
violet_dark	148:000:211_RGB	violet_blue	138:043:226_RGB
violet_purple	160:032:240_RGB	violet_thistle	216:191:216_RGB

Appears in list at level 2 containing **_display_conn_id**. Where no value is given, the assumed value is 'black'. **[display_conn]**

_display_conn_id (*numb*)

The identifying number of a display object which is connected to another display object in level 1 of the loop packet which is designated as **_display_id**. The ID number appearing as the **_display_conn_id** *must* appear elsewhere in the loop structure as a value for **_display_id**.

Appears in list at level 2 as essential element of loop structure. **Must** match parent data name **_display_id**. **[display_conn]**

_display_conn_symbol (*char*)

The symbol code for the bond connection to the displayed objects identified by **_display_id** and **_display_conn_id**. The symbol codes and descriptions are stored in the validation file 'mif_core_bonds.val'.

The data value must be one of the following:

.c	dotted line to object centres
-c	dashed line to object centres
1c	solid line to object centres
2c	double solid line to object centres
3c	triple solid line to object centres
.b	dotted line to object boundary
-b	dashed line to object boundary
1b	solid line to object boundary
2b	double solid line to object boundary
3b	triple solid line to object boundary
>b	wedge with apex at **_display_conn_id**
<b	wedge with apex at **_display_id**

Appears in list at level 2 containing **_display_conn_id**. **[display_conn]**

_display_coord_x
_display_coord_y (*numb*)

The projected coordinates of the display object identified in **_display_object** and **_display_symbol**. A display diagram has the *y* axis from bottom to top and the *x* axis from bottom left to right. The display origin is defined with **_display_origin**. Coordinate values can be scaled using **_display_scale** and **_display_span_**.

Appears in list containing **_display_id**. **[display]**

_display_define_origin (*char*)

A code signaling where the origin of the **_display_coord_** values should be placed.

The data value must be one of the following:

centre	at centre of diagram
bottom	at the bottom left corner
top	at the top left corner

Where no value is given, the assumed value is 'bottom'. **[display_define]**

_display_define_scale (*numb*)

The multiplication scale needed to convert **_display_coord_** pixels to centimetres.

The permitted range is $0. \rightarrow \infty$. **[display_define]**

_display_define_span_x
_display_define_span_y (*numb*)

The maximum dimensions of the display diagram. **_display_define_span_x** is the horizontal dimension and **_display_define_span_y** is the vertical dimension.

The permitted range is 0. → ∞. **[display_define]**

_display_id (*numb*)

The identifying number of the display object described by the combination of **_display_object** and **_display_symbol**.

Appears in list as essential element of loop structure. May match child data name(s): **_display_conn_id**. **[display]**

_display_object (*char*)

Signals how the object identified by **_display_id** is to be displayed. The value 'text' signals that **_display_symbol** contains a character string for display; the value 'icon' signals that **_display_symbol** is a code which identifies the molecular symbol to be displayed; and the value 'null' signals that no object will be shown at this site.

Appears in list containing **_display_id**.

The data value must be one of the following:

text	character string specified as **_display_symbol**
icon	icon specified as a **_display_symbol** code
.	no object is displayed

Where no value is given, the assumed value is '.'. **[display]**

_display_size (*numb*)

This value specifies the maximum *y* dimension of the object identified by **_display_symbol** in pixels.

Appears in list containing **_display_id**.

The permitted range is 0. → ∞. **[display]**

_display_symbol (*char*)

The interpretation of this item is dependent on the value of the item **_display_object**. If **_display_object** is 'text', then the **_display_symbol** is assumed to be a literal character string that should be displayed. If **_display_object** is 'icon', then the **_display_symbol** is a code that specifies the molecular symbol or icon to be displayed. The permitted code enumerations for **_display_symbol** and their descriptions are stored in the separate validation file 'mif_core_molecules.val'.

Appears in list containing **_display_id**, **_display_object**.

The data value must be one of the following:

3s	unsaturated three-membered ring with circle
4s	unsaturated four-membered ring with circle
5s	unsaturated five-membered ring with circle
6s	unsaturated six-membered ring with circle
7s	unsaturated seven-membered ring with circle
8s	unsaturated eight-membered ring with circle
lp	lone pair symbol
*	any text string (when **_display_object** is 'text')

[display]

_molecule_name_common
_molecule_name_iupac
_molecule_name_cas_8ci
_molecule_name_cas_9ci
_molecule_name_cas_10ci (*char*)

Specifies the short and systematic names for the molecule described within the data block or save frame. Data names can have the extensions

iupac	IUPAC name (IUPAC, 1971, 1979)
cas_8ci	*Chemical Abstracts 8th Collective Index*
cas_9ci	*Chemical Abstracts 9th Collective Index*
cas_10ci	*Chemical Abstracts 10th Collective Index*

References: IUPAC (1971). *Pure Appl. Chem.* **28**, 1–110; IUPAC (1979). *The Notation of Organic Chemistry.* Oxford: Pergamon.

[molecule]

_reference_conformation
_reference_stereo_group (*char*)

The frame-code pointer strings (starting with a $ character) which identify save frames containing specific categories of data which have been specified with a **_define_** data value.

Appears in list as essential element of loop structure. **[reference]**

_stereo_atom_id (*char*)

Specifies the identity of the stereogenic centre to which the sites defined under **_stereo_vertex_id** are attached.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_id**. **[stereo]**

_stereo_bond_id_1
_stereo_bond_id_2 (*char*)

Specifies the atom-site identifiers of the connected atoms that form the stereogenic bond. These atom sites are connected to the atom sites defined under **_stereo_vertex_id** and which define the stereochemistry of the cited bond.

Appears in list as essential element of loop structure. **Must** match parent data name **_atom_id**. **[stereo]**

_stereo_geometry (*char*)

Specifies the geometry of the stereocentre being described. Note that the enumeration values are not descriptions of the chemical entities, only the stereo geometry. Thus, the stereochemistry of hindered biphenyls may conform to the *allene* geometry, although they are not chemically allenes. The order of the associated **_stereo_vertex_id** values is determined by the geometry. The first site is selected as that closest to the axis of highest symmetry for the described geometry. Subsequent sites are selected sequentially by right-hand rotations about this axis; otherwise by selecting a site closest to the initial site.

Appears in list as essential element of loop structure.

The data value must be one of the following:

square	four sites
olefin	four sites in a rectangle
allene	four sites in a distorted tetrahedron
tetrahedron	four sites
square_pyramid	five sites
trigonal_bipyramid	five sites
octahedron	six sites
cube	eight sites

[stereo]

_stereo_vertex_id (*char*)

Specifies the identity of atom sites at the vertices of the stereogenic centre. The order of the selected vertex sites for each geometry is described in **_stereo_geometry**. Where one or more of the vertices in the specified geometry corresponds to an implicit hydrogen atom or unshared electron pair, the value '.' may be used. The special syntax for representing atom sites within molecular templates is described in the **_atom_id** definition.

Appears in list at level 2 as essential element of loop structure. **Must** match parent data name **_atom_id**. **[stereo]**

4.9. DDL1 dictionary

BY S. R. HALL

This is version 1.4.1 of the dictionary definition language (DDL1) that provides a machine-readable description of the attributes of data items in the core CIF and related dictionaries (Chapters 4.1 to 4.4). This version of DDL is described in Chapter 2.5.

_category (*char*)
Character string which identifies the natural grouping of data items to which the specified data item belongs. If the data item belongs in a looped list, then it must be grouped only with items from the same category, but there may be more than one looped list of the same category provided that each loop has its own independent reference item (see **_list_reference**).

[category]

_definition (*char*)
The text description of the defined item.

[definition]

_dictionary_history (*char*)
A chronological record of the changes to the dictionary file containing the definition. Normally this item is stored in the separate data block labelled **data_on_this_dictionary**.

[dictionary]

_dictionary_name (*char*)
The name string which identifies the generic identity of the dictionary. The standard construction for these names is *<application code>*_*<dictionary version>*.dic. Normally this item is stored in the separate data block labelled **data_on_this_dictionary**.
Examples: 'ddl_core.dic', 'cif_core.dic'. **[dictionary]**

_dictionary_update (*char*)
The date that the dictionary was last updated. Normally this item is stored in the separate data block labelled **data_on_this_dictionary**.

[dictionary]

_dictionary_version (*numb*)
The dictionary version number. Version numbers cannot decrease with updates. Normally this item is stored in the separate data block labelled **data_on_this_dictionary**.

[dictionary]

_enumeration (*char*)
Permitted value(s) for the defined item.
May appear in list as essential element of loop structure. **[enumeration]**

_enumeration_default (*char*)
The default value for the defined item if it is not specified explicitly. If a data value is not declared, the default is assumed to be the 'most likely' or 'natural' value.

[enumeration_default]

_enumeration_detail (*char*)
A description of the permitted value(s) for the defined item, as identified by **_enumeration**.
May appear in list containing **_enumeration**. **[enumeration]**

_enumeration_range (*char*)
The range of values permitted for a defined item. This can apply to 'numb' or 'char' items which have a preordained sequence (*e.g.* numbers or alphabetical characters). The minimum value 'min' and maximum value 'max' are separated by a colon character. If 'max' is omitted, then the item can have any permitted value greater than or equal to 'min'.
Examples: '-4:10', 'a:z', 'B:R', '0:'. **[enumeration_range]**

_example (*char*)
An example value of the defined item.
May appear in list as essential element of loop structure. **[example]**

_example_detail (*char*)
A description of an example value for the defined item.
May appear in list containing **_example**. **[example]**

_list (*char*)
Signals whether the defined item is declared in a looped list.
The data value must be one of the following:

yes	can only be declared in a looped list
no	cannot be declared in a looped list
both	declaration in a looped list optional

Where no value is given, the assumed value is 'no'. **[list]**

_list_level (*numb*)
Specifies the level of the loop structure in which a defined item with the attribute **_list** 'yes' or 'both' must be declared.
The permitted range is $1 \rightarrow \infty$. Where no value is given, the assumed value is '1'. **[list]**

_list_link_child (*char*)
Identifies data item(s) by name which must have a value which matches that of the defined item. These items are referred to as 'child' references because they depend on the existence of the defined item.
May appear in list. **[list_link_child]**

_list_link_parent (*char*)
Identifies a data item by name which must have a value which matches that of the defined item, and which must be present in the same data block as the defined item. This provides for a reference to the 'parent' data item.
May appear in list. **[list_link_parent]**

_list_mandatory (*char*)
Signals whether the defined item must be present in the loop structure containing other items of the designated **_category**. This property is transferrable to another data item which is identified by **_related_item** and has **_related_function** set as 'alternate'.
The data value must be one of the following:

yes	required item in this category of looped list
no	optional item in this category of looped list

Where no value is given, the assumed value is 'no'. **[list]**

_list_reference (*char*)
Identifies the data item, or items, which must be present (collectively) in a looped list with the defined data item for the loop structure to be valid. The data item(s) identified by **_list_reference** provide a unique access code to each loop packet. Note that this property may be transferred to another item with **_related_function** 'alternate'.
May appear in list. **[list_reference]**

Affiliation: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia.

_list_uniqueness (*char*)

Identifies data items which, collectively, must have unique values for the loop structure of the designated **_category** items to be valid. This attribute is specified in the definition of a data item with **_list_mandatory** set to 'yes'.

May appear in list. **[list_uniqueness]**

_name (*char*)

The data name(s) of the defined item(s). If data items are closely related or represent an irreducible set, their names may be declared as a looped sequence in the same definition.

May appear in list.

Examples: '_atom_site_label', '_atom_attach_all _atom_attach_ring', '_index_h _index_k _index_l', '_matrix_11 _matrix_12 _matrix_21 _matrix_22'. **[name]**

_related_function (*char*)

Specifies the relationship between the defined item and the item specified by **_related_item**. The following classifications are recognized. 'alternate' signals that the item referred to in **_related_item** has attributes that permit it to be used as an alternative to the defined item for validation purposes. 'convention' signals that the item referred to in **_related_item** is equivalent to the defined item except for a predefined convention which requires a different **_enumeration** set. 'conversion' signals that the item referred to in **_related_item** is equivalent to the defined item except that different scaling or conversion factors are applied. 'replace' signals that the item referred to in **_related_item** may be used identically to replace the defined item.

Appears in list containing **_related_item**.

The data value must be one of the following:

alternate	used alternatively for validation tests
convention	equivalent except for defined convention
conversion	equivalent except for conversion factor
replace	new definition replaces the current one

[related]

_related_item (*char*)

Identifies data item(s) which have a classified relationship to the defined data item. The nature of this relationship is specified by **_related_function**.

May appear in list as essential element of loop structure. **[related]**

_type (*char*)

The type specification of the defined item. Type 'numb' identifies items which must have values that are identifiable numbers. The acceptable syntax for these numbers is application-dependent, but the formats illustrated by the following identical numbers are considered to be interchangeable: 42, 42.000, 0.42E2, .42E+2, 4.2E1, 420000D-4, 0.0000042D+07. Type 'char' identifies items which need not be interpretable numbers. The specification of these items must comply with the STAR syntax specification of either a 'contiguous single-line string' bounded by blanks or blank-quotes, or a 'text string' bounded by semicolons as the first character of a line. Type 'null' identifies items which appear in the dictionary for data-definition and descriptive purposes. These items serve no function outside the dictionary files.

The data value must be one of the following:

numb	numerically interpretable string
char	character or text string
null	for dictionary purposes only

[type]

_type_conditions (*char*)

Codes defining conditions on the **_type** specification. 'su' permits a number string to contain an appended standard uncertainty number enclosed within parentheses, *e.g.* 4.37(5). 'esd' is a deprecated synonym for 'su', arising from the former use of the term 'estimated standard deviation' for 'standard uncertainty', and permitting a number string to contain an appended standard uncertainty within parentheses, *e.g.* 4.37(5). 'seq' permits data to be declared as a sequence of values separated by a comma <,> or a colon <:>. The sequence $v_1, v_2, v_3 \ldots$ signals that v_1, v_2, v_3 *etc.* are alternative values. The sequence v_1:v_2 signals that v_1 and v_2 are the boundary values of a continuous range of values satisfying the requirements of **_enumeration** for the defined item. Combinations of alternative and range sequences are permitted.

May appear in list.

The data value must be one of the following:

none	no extra conditions apply to the defined **_type**
esd	synonym for 'su'
seq	data may be declared as a permitted sequence
su	numbers *may* have s.u.'s appended within parentheses

Where no value is given, the assumed value is 'none'. **[type_conditions]**

_type_construct (*char*)

String of characters specifying the construction of the data value for the defined data item. The construction is composed of two entities: (1) data names; (2) construction characters. The rules of construction conform to the regular expression (REGEX) specifications detailed in IEEE (1991) and *International Tables for Crystallography* (2005), Volume G, Chapter 2.5.

References: IEEE (1991). *IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and Utilities*, Vol. 1, IEEE Standard 1003.2-1992. New York: The Institute of Electrical Engineers. *International Tables for Crystallography* (2005). Vol. G, *Definition and Exchange of Crystallographic Data*, edited by S. R. Hall and B. McMahon. Heidelberg: Springer.

Where no value is given, the assumed value is '.*'.

Example: '(_year)-(_month)-(_day)' (a typical construction for _date). **[type_construct]**

_units (*char*)

A unique code which identifies the units of the defined data item. A description of the units is provided in **_units_detail**.

Examples: 'K' (kelvins), 'C' (degrees Celsius), 'rad' (radians), 'e' (electrons), 'V' (volts), 'Dal' (daltons), 'm' (metres), 'kg' (kilograms), 's' (seconds). **[units]**

_units_detail (*char*)

A description of the numerical units applicable to the defined item and identified by the code **_units**.

[units]

4.10. DDL2 dictionary

BY J. D. WESTBROOK AND S. R. HALL

This is version 2.1.3 of the dictionary definition language (DDL2) that provides a machine-readable description of the attributes of data items in the mmCIF and related dictionaries (Chapters 4.5 to 4.7). This version of DDL is described in Chapter 2.6.

The category groups in the DDL2 dictionary are: ddl_group (component categories of the macromolecular DDL); data-block_group (categories that describe the characteristics of data blocks); category_group (categories that describe the characteristics of categories); sub_category_group (categories that describe the characteristics of subcategories); item_group (categories that describe the characteristics of data items); dictionary_group (categories that describe the dictionary); and compliance_group (categories that are retained specifically for compliance with older versions of the DDL).

CATEGORY

Attributes defining the functionality for the entire category.
Category group(s): **ddl_group**
category_group
Category key(s): **_category.id**

***_category.description** (*text*)
Text description of a category.

[category]

***_category.id** (*idname*)
The identity of the data category. Data items may only be looped with items of the same category.
The following item(s) have an equivalent role in their respective categories:
_category_examples.id,
_category_group.category_id,
_category_key.id,
_category_methods.category_id,
_item.category_id. **[category]**

_category.implicit_key
An identifier that may be used to distinguish the contents of like categories between data blocks.

***_category.mandatory_code** (*code*)
Whether the category must be specified in a dictionary.

[category]

CATEGORY_EXAMPLES

Example applications and descriptions of data items in this category.
Category key(s): **_category_examples.id**
_category_examples.case

***_category_examples.case** (*text*)
A case of examples involving items in this category.

[category_examples]

_category_examples.detail (*text*)
A description of an example given in **_category_examples.case**.

[category_examples]

CATEGORY_GROUP

Provides a list of category groups to which the base category belongs.
Category group(s): **ddl_group**
category_group
Category key(s): **_category_group.id**
_category_group.category_id

CATEGORY_GROUP_LIST

This category provides the definition of each category group. A category group is a collection of related categories.
Category group(s): **ddl_group**
category_group
Category key(s): **_category_group_list.id**

***_category_group_list.description** (*text*)
Text description of a category group.

[category_group_list]

***_category_group_list.id** (*idname*)
The name of a category group.
The following item(s) have an equivalent role in their respective categories:
_category_group.id,
_category_group_list.parent_id. **[category_group_list]**

_category_group_list.parent_id
The name of the optional parent category group.

CATEGORY_KEY

This category holds a list of the item names that uniquely identify the elements of the category.
Category group(s): **ddl_group**
category_group
Category key(s): **_category_key.name**
_category_key.id

_category_key.name
The name of a data item that serves as a key identifier for the category (*e.g.* a component of the primary key).

CATEGORY_METHODS

Attributes specifying the association between categories and methods.
Category group(s): **ddl_group**
category_group
Category key(s): **_category_methods.method_id**
_category_methods.category_id

Affiliations: JOHN D. WESTBROOK, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ, USA; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia.

DATABLOCK

Attributes defining the characteristics of a data block.
Category group(s): **ddl_group**
datablock_group
Category key(s): **_datablock.id**

*__datablock.description__ (*text*)
Text description of the data block.
[datablock]

(*)__datablock.id__ (*code*)
The identity of the data block.
The following item(s) have an equivalent role in their respective categories:
_datablock_methods.datablock_id,
_dictionary.datablock_id,
_category.implicit_key. **[datablock]**

DATABLOCK_METHODS

Attributes specifying the association between data blocks and methods.
Category group(s): **ddl_group**
datablock_group
Category key(s): **_datablock_methods.method_id**
_datablock_methods.datablock_id

DICTIONARY

Attributes for specifying the dictionary title, version and data-block identifier.
Mandatory category.
Category group(s): **ddl_group**
datablock_group
dictionary_group
Category key(s): **_dictionary.datablock_id**

__dictionary.datablock_id__
The identifier for the data block containing the dictionary.

*__dictionary.title__ (*char*)
Title identifier of the dictionary.
[dictionary]

__dictionary.version__
A unique version identifier for the dictionary.

DICTIONARY_HISTORY

Attributes for specifying the revision history of the dictionary.
Category group(s): **ddl_group**
dictionary_group
Category key(s): **_dictionary_history.version**

*__dictionary_history.revision__ (*text*)
Text description of the dictionary revision.
[dictionary_history]

*__dictionary_history.update__ (*yyyy-mm-dd*)
The date that the last dictionary revision took place.
[dictionary_history]

*__dictionary_history.version__ (*char*)
A unique version identifier for the dictionary revision.
The following item(s) have an equivalent role in their respective categories:
_dictionary.version. **[dictionary_history]**

ITEM

Attributes which describe the characteristics of a data item.
Category group(s): **ddl_group**
item_group
Category key(s): **_item.name**

*__item.mandatory_code__ (*code*)
Signals whether the defined item is mandatory for the proper description of its category.
The data value must be one of the following:

yes	required item in this category
no	optional item in this category
implicit	required item but may be determined from context

[item]

(*)__item.name__ (*name*)
Data name of the defined item.
The following item(s) have an equivalent role in their respective categories:
_category_key.name,
_item_aliases.name,
_item_default.name,
_item_dependent.name,
_item_dependent.dependent_name,
_item_description.name,
_item_enumeration.name,
_item_examples.name,
_item_linked.child_name,
_item_linked.parent_name,
_item_methods.name,
_item_range.name,
_item_related.name,
_item_related.related_name,
_item_type.name,
_item_type_conditions.name,
_item_structure.name,
_item_sub_category.name,
_item_units.name. **[item]**

ITEM_ALIASES

This category holds a list of possible alias names or synonyms for each data item. Each alias name is identified by the name and version of the dictionary to which it belongs.
Category key(s): **_item_aliases.alias_name**
_item_aliases.dictionary
_item_aliases.version

*__item_aliases.alias_name__ (*aliasname*)
Alias name for the data item.
[item_aliases]

*__item_aliases.dictionary__ (*char*)
The dictionary in which the alias name is defined.
[item_aliases]

*__item_aliases.version__ (*char*)
The version of the dictionary in which the alias name is defined.
[item_aliases]

ITEM_DEFAULT

Attributes specifying the default value for a data item.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_default.name**

_item_default.value (*any*)
The default value for the defined item if it is not specified explicitly. If a data value is not declared, the default is assumed to be the most likely or natural value.

[item_default]

ITEM_DEPENDENT

Attributes which identify other data items that must be specified for the defined data item to be valid.
Category key(s): **_item_dependent.name**
_item_dependent.dependent_name

_item_dependent.dependent_name
Data name of a dependent item.

ITEM_DESCRIPTION

This category holds the descriptions of each data item.
Mandatory category.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_description.name**
_item_description.description

*__item_description.description__ (*text*)
Text description of the defined data item.

[item_description]

ITEM_ENUMERATION

Attributes which specify the permitted enumeration of the items.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_enumeration.name**
_item_enumeration.value

_item_enumeration.detail (*text*)
A description of a permissible value for the defined item.

[item_enumeration]

*__item_enumeration.value__ (*any*)
A permissible value, character or number for the defined item.

[item_enumeration]

ITEM_EXAMPLES

Attributes for describing examples of applications of the data item.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_examples.name**
_item_examples.case

_item_examples.case (*text*)
An example application of the defined data item.

[item_examples]

_item_examples.detail (*text*)
A description of an example specified in **_item_examples.case**.

[item_examples]

ITEM_LINKED

Attributes which describe how equivalent data items are linked within categories and across different categories.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_linked.child_name**

_item_linked.child_name
Name of the child data item.

_item_linked.parent_name
Name of the parent data item.

ITEM_METHODS

Attributes specifying the association between data items and methods.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_methods.method_id**
_item_methods.name

ITEM_RANGE

The range of permissible values of a data item. When multiple ranges are specified, they are interpreted sequentially using a logical OR. To specify that an item value may be equal to a boundary value, specify an item range where the maximum and mimimum values equal the boundary value.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_range.name**
_item_range.minimum
_item_range.maximum

_item_range.maximum (*any*)
Maximum permissible value of a data item or the upper bound of a permissible range (maximum value $>$ data value).

[item_range]

_item_range.minimum (*any*)
Minimum permissible value of a data item or the lower bound of a permissible range (minimum value $<$ data value).

[item_range]

ITEM_RELATED

Attributes which specify recognized relationships between data items.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_related.name**
_item_related.related_name
_item_related.function_code

*`_item_related.function_code` (*code*)

The code for the type of relationship between the item identified by `_item_related.name` and the defined item.

'alternate' indicates that the item identified in `_item_related.related_name` is an alternative expression in terms of its application and attributes to the item in this definition. 'alternate_exclusive' indicates that the item identified in `_item_related.related_name` is an alternative expression in terms of its application and attributes to the item in this definition. Only one of the alternative forms may be specified.

'convention' indicates that the item identified in `_item_related.related_name` differs from the defined item only in terms of a convention in its expression.

'conversion_constant' indicates that the item identified in `_item_related.related_name` differs from the defined item only by a known constant. 'conversion_arbitrary' indicates that the item identified in `_item_related.related_name` differs from the defined item only by an arbitrary constant.

'replaces' indicates that the defined item replaces the item identified in `_item_related.related_name`. 'replacedby' indicates that the defined item is replaced by the item identified in `_item_related.related_name`.

'associated_value' indicates that the item identified in `_item_related.related_name` is meaningful when associated with the defined item. 'associated_esd' indicates that the item identified in `_item_related.related_name` is the estimated standard deviation of the defined item. 'associated_error' indicates that the item identified in `_item_related.related_name` is the estimated error of the defined item.

The data value must be one of the following:

`alternate`	alternate form of the item
`alternate_exclusive`	mutually exclusive alternate form of the item
`convention`	depends on defined convention
`conversion_constant`	related by a known conversion factor
`conversion_arbitrary`	related by an arbitrary conversion factor
`replaces`	a replacement definition
`replacedby`	an obsolete definition
`associated_value`	a meaningful value when related to the item
`associated_esd`	an estimated standard deviation of the item
`associated_error`	an estimated error of the item

[item_related]

ITEM_STRUCTURE

This category holds the association between data items and named vector/matrix declarations.

Category group(s): **ddl_group**
item_group
Category key(s): **_item_structure.name**

*`_item_structure.organization` (*code*)

Identifies whether the structure is defined in column- or row-major order. Only the unique elements of symmetric matrices are specified.

The data value must be one of the following:

`columnwise`	column-major order
`rowwise`	row-major order

ITEM_STRUCTURE_LIST

This category holds a description for each structure type.

Category group(s): **ddl_group**
item_group
Category key(s): **_item_structure_list.code**
_item_structure_list.index

*`_item_structure_list.code` (*code*)

The name of the matrix/vector structure declaration.

The following item(s) have an equivalent role in their respective categories:

_item_structure.code. **[item_structure_list]**

*`_item_structure_list.dimension` (*int*)

Identifies the length of this row or column of the structure.

The permitted range is $[1, \infty)$. **[item_structure_list]**

*`_item_structure_list.index` (*int*)

Identifies the one-based index of a row or column of the structure.

The permitted range is $[1, \infty)$. **[item_structure_list]**

ITEM_SUB_CATEGORY

This category assigns data items to subcategories.

Category group(s): **sub_category_group**
item_group
Category key(s): **_item_sub_category.id**
_item_sub_category.name

ITEM_TYPE

Attributes for specifying the data-type code for each data item.

Category group(s): **ddl_group**
item_group
Category key(s): **_item_type.name**

ITEM_TYPE_CONDITIONS

Attributes for specifying additional conditions associated with the data type of the item.

Category group(s): **ddl_group**
item_group
compliance_group
Category key(s): **_item_type_conditions.name**

*`_item_type_conditions.code` (*code*)

Codes defining conditions on the `_item_type.code` specification. 'esd' permits a number string to contain an appended standard deviation number enclosed within parentheses, *e.g.* 4.37(5). 'seq' permits data to be declared as a sequence of values separated by a comma <,> or a colon <:>. The sequence $v_1, v_2, v_3 \ldots$ signals that v_1, v_2, v_3 *etc.* are alternative values for the data item. The sequence v_1:v_2 signals that v_1 and v_2 are the boundary values of a continuous range of values. This mechanism was used to specify permitted ranges of an item in previous DDL versions. Combinations of alternate and range sequences are permitted.

The data value must be one of the following:

`none`	no extra conditions apply to this data item
`esd`	numbers may have esd values appended within parentheses
`seq`	data may be declared as a comma- or colon-separated sequence

[item_type_conditions]

ITEM_TYPE_LIST

Attributes which define each type code.

Category group(s): **ddl_group**
item_group
Category key(s): **_item_type_list.code**

*`_item_type_list.code` (*code*)

The codes specifying the nature of the data value.

The following item(s) have an equivalent role in their respective categories:

_item_type.code. **[item_type_list]**

_item_type_list.construct (*text*)
When a data value can be defined as a pre-determined sequence of characters, optional characters or data names (for which the definition is also available), it is specified as a construction. The rules of construction conform to the the regular expression (REGEX) specifications detailed in IEEE (1991). Resolved data names for which **_item_type_list.construct** specifications exist are replaced by these constructions, otherwise the data-name string is not replaced.

Reference: IEEE (1991). *IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and Utilities*, Vol. 1, IEEE Standard 1003.2-1992. New York: The Institute of Electrical Engineers.

Example: '_year-_month-_day' (typical construction for _date). **[item_type_list]**

_item_type_list.detail (*text*)
An optional description of the data type.

[item_type_list]

***_item_type_list.primitive_code** (*code*)
The codes specifying the primitive type of the data value.

The data value must be one of the following:

numb	numerically interpretable string
char	character or text string (case-sensitive)
uchar	character or text string (case-insensitive)
null	for dictionary purposes only

[item_type_list]

ITEM_UNITS

Specifies the physical units in which data items are expressed.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_units.name**

ITEM_UNITS_CONVERSION

Conversion factors between the various units of measure defined in the ITEM_UNITS_LIST category.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_units_conversion.from_code**
_item_units_conversion.to_code

***_item_units_conversion.factor** (*any*)
The arithmetic operation required to convert between the unit systems: ⟨to_code⟩ = ⟨from_code⟩⟨operator⟩⟨factor⟩.

[item_units_conversion]

_item_units_conversion.from_code
The unit system on which the conversion operation is applied to produce the unit system specified in **_item_units_conversion.to_code**: ⟨to_code⟩ = ⟨from_code⟩⟨operator⟩⟨factor⟩.

***_item_units_conversion.operator** (*code*)
The arithmetic operator required to convert between the unit systems: ⟨to_code⟩= ⟨from_code⟩⟨operator⟩⟨factor⟩.

The data value must be one of the following:

+	addition
-	subtraction
*	multiplication
/	division

[item_units_conversion]

_item_units_conversion.to_code
The unit system produced after an operation is applied to the unit system specified by **_item_units_conversion.from_code**: ⟨to_code⟩ = ⟨from_code⟩⟨operator⟩⟨factor⟩.

ITEM_UNITS_LIST

Attributes which describe the physical units of measurement in which data items may be expressed.
Category group(s): **ddl_group**
item_group
Category key(s): **_item_units_list.code**

***_item_units_list.code** (*code*)
The code specifying the name of the unit of measurement.

The following item(s) have an equivalent role in their respective categories:

_item_units.code,
_item_units_conversion.from_code,
_item_units_conversion.to_code. **[item_units_list]**

_item_units_list.detail (*text*)
A description of the unit of measurement.

[item_units_list]

METHOD_LIST

Attributes specifying the list of methods applicable to data items, subcategories and categories.
Category group(s): **ddl_group**
item_group
category_group
Category key(s): **_method_list.id**

***_method_list.code** (*code*)
A code that describes the function of the method.

Examples: 'calculation' (method to calculate the item), 'verification' (method to verify the data item), 'cast' (method to provide cast conversion), 'addition' (method to define item + item), 'division' (method to define item / item), 'multiplication' (method to define item × item), 'equivalence' (method to define item = item), 'other' (miscellaneous method). **[method_list]**

_method_list.detail (*text*)
Description of application method in **_method_list.id**.

[method_list]

***_method_list.id** (*idname*)
Unique identifier for each method listed.

The following item(s) have an equivalent role in their respective categories:

_item_methods.method_id,
_category_methods.method_id,
_sub_category_methods.method_id,
_datablock_methods.method_id. **[method_list]**

***_method_list.inline** (*text*)
In-line text of a method associated with the data item.

[method_list]

***_method_list.language** (*code*)
Language in which the method is expressed.

Examples: 'BNF', 'C', 'C++', 'FORTRAN', 'LISP', 'PASCAL', 'PERL', 'TCL', 'OTHER'.

[method_list]

SUB_CATEGORY

The purpose of a subcategory is to define an association between data items within a category and optionally to provide a method to validate the collection of items. For example, the subcategory named 'cartesian' might be applied to the data items for the coordinates *x*, *y* and *z*.
Category group(s): **ddl_group**
sub_category_group
Category key(s): **_sub_category.id**

*`_sub_category.description` (*text*)
Description of the subcategory.

`[sub_category]`

*`_sub_category.id` (*idname*)
The identity of the subcategory.

The following item(s) have an equivalent role in their respective categories:
`_sub_category_examples.id`,
`_sub_category_methods.sub_category_id`,
`_item_sub_category.id`. `[sub_category]`

*`_sub_category_examples.case` (*text*)
An example involving items in this subcategory.

`[sub_category_examples]`

`_sub_category_examples.detail` (*text*)
A description of an example given in `_sub_category_examples.case`.

`[sub_category_examples]`

SUB_CATEGORY_EXAMPLES

Example applications and descriptions of data items in this subcategory.

Category group(s): `ddl_group`
`sub_category_group`
Category key(s): `_sub_category_examples.id`
`_sub_category_examples.case`

SUB_CATEGORY_METHODS

Attributes specifying the association between subcategories and methods.

Category group(s): `ddl_group`
`sub_category_group`
Category key(s): `_sub_category_methods.method_id`
`_sub_category_methods.sub_category_id`

5. APPLICATIONS

BY H. M. BERMAN, H. J. BERNSTEIN, P. J. ELLIS, Z. FENG, S. R. HALL, M. A. HOYLAND, B. MCMAHON, N. SPADACCINI, P. R. STRICKLAND, J. D. WESTBROOK AND H. YANG

5.1. General considerations in programming CIF applications (H. J. BERNSTEIN) 481
5.1.1. Introduction 481
5.1.2. Background 481
5.1.3. Strategies in designing a CIF-aware application 483
5.1.4. Conclusion 486
References 486
5.2. STAR File utilities (N. SPADACCINI, S. R. HALL AND B. MCMAHON) 488
5.2.1. Introduction 488
5.2.2. Data instances and context 488
5.2.3. *Star_Base*: a general-purpose data extractor for STAR Files 491
5.2.4. Editing STAR Files with *Star.vim* 494
5.2.5. Browser-based viewing with *StarMarkUp* 494
5.2.6. Object-oriented STAR programming 495
References 497
5.3. Syntactic utilities for CIF (B. MCMAHON) 499
5.3.1. Introduction 499
5.3.2. Syntax checker 499
5.3.3. Editors with graphical user interfaces 501
5.3.4. Data-name validation 507
5.3.5. File transformation software 509
5.3.6. Libraries for scripting languages 515
5.3.7. Rapid development tools 517
5.3.8. Tools for mmCIF 522
5.3.9. Concluding remarks 525
References 525
5.4. *CIFtbx*: Fortran tools for manipulating CIFs (H. J. BERNSTEIN AND S. R. HALL) 526
5.4.1. Introduction 526
5.4.2. An overview of the library 526
5.4.3. Initialization commands 526
5.4.4. Read commands 527
5.4.5. Write commands 527
5.4.6. Variables 528
5.4.7. Name aliases 529
5.4.8. Implementation of the tools 530
5.4.9. How to read CIF data 530
5.4.10. How to write a CIF 530

5.4.11. Error-message glossary ... 531
5.4.12. Internals and programming style ... 535
5.4.13. Distribution ... 538
References ... 538
5.5. The use of mmCIF architecture for PDB data management (J. D. WESTBROOK, H. YANG, Z. FENG AND H. M. BERMAN) ... 539
5.5.1. Introduction ... 539
5.5.2. Representing macromolecular structure data ... 539
5.5.3. Integrated data-processing system: overview ... 541
5.5.4. Access ... 543
References ... 543
5.6. *CBFlib*: an ANSI C library for manipulating image data (P. J. ELLIS AND H. J. BERNSTEIN) ... 544
5.6.1. Introduction ... 544
5.6.2. *CBFlib* function descriptions ... 545
5.6.3. Compression schemes ... 552
5.6.4. Sample templates ... 552
5.6.5. Example programs ... 555
References ... 556
5.7. Small-molecule crystal structure publication using CIF (P. R. STRICKLAND, M. A. HOYLAND AND B. MCMAHON) ... 557
5.7.1. Introduction ... 557
5.7.2. Case study: the fully automated reporting of small-unit-cell crystal structures ... 557
5.7.3. CIF and other journals ... 562
Appendix 5.7.1. Request list for *Acta Crystallographica Section C* ... 564
Appendix 5.7.2. Data validation using *checkcif* ... 566
References ... 568

5.1. General considerations in programming CIF applications

BY H. J. BERNSTEIN

5.1.1. Introduction

There are many ways to create new 'CIF-aware' applications and to adapt existing applications to make them CIF-aware. This chapter reviews general considerations in programming CIF-aware applications, ranging from leaving an application CIF-unaware and relying on external filter utilities to do the job, through engineering an existing application to directly read and write CIFs, to writing a new CIF-aware application from scratch. The adaptation of applications to CIF does not happen in isolation. There are many other data representations and metadata frameworks relevant to crystallography. In Chapter 1.1, the CIF format was placed in the historical context of the development of data representation and metadata languages. In this chapter, we deal with that context from the perspective of software design.

The major issues in making an application CIF-aware are:

(1) Are CIFs to be read?
  (i) Are these CIFs produced externally?
  (ii) Does the organization of information required by the application conform to the organization of information specified in the relevant dictionaries?
  (iii) Is maximal performance in reading important?

(2) Are CIFs to be written?
  (i) Are these CIFs to be used externally?
  (ii) Does the organization of information used by the application internally conform to the organization of information specified by the relevant dictionaries?
  (iii) Is maximal performance in writing important?

Reading a CIF is a much more complex task than writing a CIF. Two equally valid CIF presentations of exactly the same information may be ordered differently. An application that reads a CIF must be prepared to accept tags and columns of tables in any order. The same order independence means that an application may write out tags and columns of tables in any convenient order, simplifying the design of CIF write logic.

When CIFs are only to be used internally, it is tempting to adjust the format to fit the application, *e.g.* by imposing order dependence. However, caution is advised if there is any possibility that such an application might eventually have to deal with CIFs coming from or going to other groups. In designing the CIF interface for an application, it is prudent to assume that the read logic will eventually have to deal with externally produced CIFs and that CIFs produced by the write logic will eventually be processed by software at other sites.

If performance is not a major issue, then it can be easy to make an application CIF-aware simply by the use of external filter programs. However, when performance *is* an issue, it becomes necessary to integrate the CIF reading and writing logic with the application. This can be done with reasonable efficiency by the use of existing CIF-aware libraries, but such libraries can impose a cost by their use of private internal data structures to hold the information from a CIF. Integrated design from scratch may be needed for maximal performance.

Creating or adapting software for CIF is an example of creating or adapting software for an agreed format, a format to be adhered to in the creation of multiple applications. There are different levels of 'agreement'. The agreement may apply to the representation of data, the representation of information about data ('metadata') or both (making a 'data framework'). The agreement may loosely specify the style of presentation of information or may specify details of presentation with great precision, or anything in between. The effort one needs to make in adapting an application to an agreed format depends to a large part on the level of agreement. If the agreed format is sufficiently detailed, one can comply strictly with the agreed format as a standard.

If one's goals are the widest possible interchange of data and the longest possible survival time of data in archives, it is important to achieve strict adherence to the agreed format as a standard. If one's goals are shorter-term and do not raise issues of interchange with independent groups, use of an agreed format as a checklist or style guide may help to avoid redundant effort. Even when one has longer-term goals, system requirements may not mesh with the agreed format and the development of new formats may be needed. CIF, as an agreed format, involves specification of metadata for a specific data format and of ontologies of domain-specific definitions that could be used not only in CIF but also in other formats. The use of an agreed format as a base can help to avoid redundant effort in this case as well. Within the domain of small-molecule crystallography, CIF has achieved its powerful impact on crystallographic publication and archiving by being used as a strict standard both for the data format and for definitions of terms. Within other domains or for certain applications other approaches may be appropriate. For example, it has proven productive to make use of CIF data names within XML (Bray *et al.*, 1998) formatted documents (Bernstein & Bernstein, 2002).

5.1.2. Background

There have been many efforts at creating agreed formats for data to be used in crystallography (see Chapter 1.1). We need to consider how software has been created to make use of such formats, especially software to make use of CIF.

Agreement on formats evolved from the earliest efforts at collaboration among research groups. Within crystallography, recognition of the need to use data formats as standards and to adapt applications to agreed formats, rather than to adapt formats to the caprices of particular applications or diffractometers or graphics engines, began in the late 1960s and early 1970s with the establishment of computerized data resources for the chemical and crystallographic community and the increasing availability of computer networks (Lykos, 1975). We will discuss three early data-resource efforts: the Cambridge Crystallographic Data Centre Structural Database File (CSD) (Allen *et al.*, 1973), the Brookhaven National Laboratory Protein Data Bank (PDB) (Bernstein *et al.*, 1977) and the NIH/EPA Chemical Information System (CIS) (Heller *et al.*, 1977). The differences and similarities among application development efforts related to these resources illustrate some of the issues

Affiliation: HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA.

that now face software developers working with CIF: conformance to agreed formats *versus* deviations from standards to improve performance, as well as cross-platform portability.

The Cambridge Crystallographic Data Centre was established in 1965 'to compile a database containing comprehensive information on small-molecule crystal structures, *i.e.* organics and metallo-organic compounds containing up to 500 non-H atoms, the structures of which had been determined by X-ray or neutron diffraction' (Allen, 2002). The Protein Data Bank was established at Brookhaven National Laboratory in 1971 as an archive of macromolecular structural information. The NIH/EPA Chemical Information System was established in 1975 as a confederation of databases including mass spectroscopy, NMR and the data from the CSD. The three resources, CSD, PDB and CIS, took different approaches to applications development. The CSD was an integrated software system centred on a database. Both the software and the database were distributed on magnetic tape for users to use on their local computers. The developers of the software had to be concerned with portability of the software across the multiple computer systems used by crystallographers, but retained control of the design of the retrieval software and a core suite of applications. The PDB was an archive, rather than a database. Some software and the data were distributed on magnetic tape, but the application development model was what would now be called 'open', with users and software developers taking the data and the PDB format specification and creating software that would do useful things with PDB entries. The CIS was a remotely accessed confederation of databases on a central computer. The developers of software for the CIS did not have to be concerned with cross-platform portability, or with changes in syntax or semantics of data files impacting on external software developers. Developers of software for the CSD and the PDB had to be concerned with strict compliance with the rules for the respective data formats, albeit on somewhat different timescales. Developers of software for the centralized CIS database could negotiate for immediate changes in the data format to improve performance of the relevant application.

The CSD had agreed internal formats (Cambridge Structural Database, 1978). However, as noted in Chapter 1.1, there were many different formats in use for small-molecule crystallography and related fields. One may conjecture that one of many causes for such divergence was the CCDC practice of acquiring much of its data from journals, after differences among data formats had been masked by the publication process. The transition from this Tower of Babel to CIF is described in Chapter 1.1, and that history will not be repeated here, but it is important to note that an application writer working in the domain of small-molecule crystallography still has to be aware of a wide variety of formats in addition to CIF.

In the beginning, the PDB went through a relatively rapid format change and then achieved a stable format for more than two decades. The PDB differed from the CSD in depending on user deposition of data prior to publication. The better a user conformed to PDB data-format conventions, the more efficiently could the data move from deposition to release. The initial standard PDB format (PDB, 1974) was derived from the format used in a popular refinement program of the day (Diamond, 1971) and used 132-character records identified by the character strings in the first six columns. Starting in 1976, the PDB spent more than a year (PDB, 1976*a*,*b*, 1977) converting to an 80-column format, extensions of which are still in use to this day. Many external programs were developed using this 80-column format and it has become a major *de facto* standard for macromolecular software applications. Most application packages producing crystallographic macromolecular structures made a gradual transition from having output options for producing 'Diamond format' to having output options for producing PDB format. Macromolecular applications working with other disciplines shared the small-molecule applications penchant for multiple formats.

The CIS, working in a completely closed, central service environment, had little direct impact on the formats to be used for applications. The CIS would acquire data from existing archives and databases and meld them into its master database. It would deliver its data as text on a CRT. Much of the impact of CIS data formats was to be restricted to its own internal application development.

Most of the formats resulting from these early efforts were fixed-field, fixed-order formats. The result was that adapting an application to a data format was simple if the processing flow of the application conformed to the fixed order of the data format. Frequently, the data flow did conform. When the processing flow did not conform, it was necessary to create internal data structures or temporary files to allow the unfortunately timed arrival of data to be time-shifted until it was needed. In general, the heaviest burden was imposed on applications that needed to write data conforming to one of the agreed formats. As the complexity of such time-shifting processes increased, it became clear that the cleanest solution was to base an application on an internal database and to populate the database as the data were processed. When data were to be written by an application, the data could be extracted from the database in whatever order was required.

In the 1970s and early 1980s, such a procedure was a serious burden to place on an application. With limited memory and processor speeds, there was a strong argument for adapting agreed formats to the 'natural' processing flow, reducing or avoiding the need for an internal database. As the speed and size of computers have changed and as programming language and operating-system support for dynamic allocation of resources has improved, the need to have agreed formats driven by applications has become less pressing.

We need to understand three major thrusts in data representation: the development of markup languages, of data-representation frameworks and of database application support. Modern applications can benefit from all three.

5.1.2.1. Markup languages

A markup language allows the raw text of a document to be annotated with interleaved 'markup' specifying layout information for the bracketed text. For document processing, the implicit assumption of the use of an internal database became formalized with the gradual adoption of agreed markup languages in the late 1980s and early 1990s [*e.g.* TEX (Knuth, 1986), SGML (ISO, 1986), RTF (Andrews, 1987), HTML (Berners-Lee, 1989)]. When used in this manner, such a language has the implicit ordering assumption of reading forward in the document. However, with modern demands for multidimensional layout and document reflow, applications managing such documents achieve the best performance and flexibility when they store the entire marked-up document in an internal data structure that allows random access to all the information.

5.1.2.2. Data-representation frameworks

A data-representation framework provides the concepts for managing data and data about the management of data ('metadata'). Such frameworks may be based on programming languages or markup languages or built from scratch. They provide a mechanism for representing data (*e.g.* as data sets, graphs or trees)

and a mechanism for representing metadata (*e.g.* as dictionaries or schemas). Four are of particular importance in crystallography: CIF, ASN.1, HDF and XML.

As noted in Chapter 1.1, CIF was created to rationalize the publication process for small molecules. It combines a very simple tag–value data representation with a dictionary definition language (DDL) and well populated dictionaries. CIF is table-oriented, naturally row-based, has case-insensitive tags and allows two levels of nesting. CIF is order-independent and uses its dictionaries both to define the meanings of its tags and to parameterize its tags. It is interesting to note that, even though CIF is defined as order-independent, it effectively fills the role of an order-dependent markup language in the publication process. We will discuss this issue later in this chapter.

Abstract Syntax Notation One (ASN.1) (Dubuisson, 2000; ISO, 2002) was developed to provide a data framework for data communications, where great precision in the bit-by-bit layout of data to be seen by very different systems is needed. Although targeted for communications software, ASN.1 is suitable for any application requiring precise control of data structures and, as such, primarily supports the metadata of an application, rather than the data. ASN.1 can be compiled directly to C code. The resulting C code then supports the data of the application. ASN.1 notation found application in NCBI's macromolecular modelling database (Ohkawa *et al.*, 1995). ASN.1 has case-sensitive tags and allows case-insensitive variants. It manages order-dependent data structures in a mixed order-dependent/order-independent environment.

HDF (NCSA, 1993) is 'a machine-independent, self-describing, extendible file format for sharing scientific data in a heterogeneous computing environment, accompanied by a convenient, standardized, public domain I/O library and a comprehensive collection of high quality data manipulation and analysis interfaces and tools' (http://ssdoo.gsfc.nasa.gov/nost/formats/hdf.html). HDF was adopted by the Neutron and X-ray Data Format (NeXus) effort (Klosowski *et al.*, 1997). HDF allows the building of a complete data framework, representing both data and metadata. Two parallel threads of software development, focused on the management and exchange of raw data from area detectors, began in the mid-1990s: the Crystallographic Binary File (CBF) (Hammersley, 1997) and NeXus. The volumes of data involved were daunting and efficiency of storage was important. Therefore both proposed formats assumed a binary format. CBF was based on a combination of CIF-like ASCII headers with compressed binary images. NeXus was based on HDF. The first API for CBF was produced by Paul Ellis in 1998. CBF rapidly evolved into CBF/imgCIF with a complete DDL2 dictionary and a fully CIF-compliant API (Chapter 5.6). As of mid-2004, NeXus was still evolving (see http://www.nexus.anl.gov/).

XML is a simplified form of SGML, drawing on years of development of tools for SGML and HTML. XML is tree-oriented with case-sensitive entity names. It allows unlimited nesting and is order-dependent. Metadata are managed as a 'document type definition' (DTD), which provides minimal syntactic information, or as schemas, which allow for more detail and are more consistent with database conventions. In fields close to crystallography, the first effort at adopting XML was the chemical markup language (CML) (Murray-Rust & Rzepa, 1999). CML is intentionally imprecise in its ontology to allow for flexibility in development. The CSD and PDB have released their own XML representations (http://www.ccdc.cam.ac.uk/support/prods_doc/relibase/Sysdocn1119.html; http://www.rcsb.org/pdb/newsletter/2003q2/xml_beta.html).

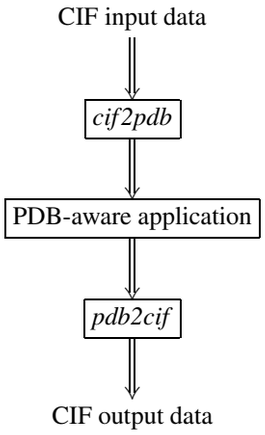


Fig. 5.1.3.1. Example of using filters to make a PDB-aware application CIF-aware.

It may seem from this discussion that the application designer faces an unmanageable variety of data frameworks in an unstable, evolving environment. To some extent this is true. Fortunately, however, there are signs of convergence on CIF dictionary-based ontologies and the use of transliterated CIFs. This means that an application adapted to CIF should be relatively easy to adapt to other data frameworks.

5.1.3. Strategies in designing a CIF-aware application

There are multiple strategies to consider when designing a CIF-aware application. One can use external filters. One can use existing CIF-aware libraries. One can write CIF-aware code from scratch.

5.1.3.1. Working with filter utilities

One solution to making an existing application aware of a new data format is to leave the application unchanged and change the data instead. For almost all crystallographic formats other than CIF, the Swiss-army knife of conversion utilities is *Babel* (Walters & Stahl, 1994). *Babel* includes conversions to and from PDB format. Therefore, by the use of *cif2pdb* (Bernstein & Bernstein, 1996) and *pdb2cif* (Bernstein *et al.*, 1998) combined with *Babel*, many macromolecular applications can be made CIF-aware without changing their code (see Figs. 5.1.3.1 and 5.1.3.2). If the need is to extract mmCIF data from the output of a major application, the PDB provides *PDB_EXTRACT* (http://sw-tools.pdb.org/apps/PDB_EXTRACT/).

Creating a filter program to go from almost any small-molecule format to core CIF is easy. In many cases one need only insert the appropriate '`loop_`' headers. Creating a filter to go from CIF to a particular small-molecule format can be more challenging, because a CIF may have its data in any order. This can be resolved by use of *QUASAR* (Hall & Sievers, 1993) or *cif2cif* (Bernstein, 1997), which accept request lists specifying the order in which data are to be presented (see Fig. 5.1.3.3).

There are a significant and growing number of filter programs available. Several of them [*QUASAR*, *cif2cif*, *ciftex* (ftp://ftp.iucr.org/pub/ciftex.tar.Z) (to convert from CIF to TEX) and ZINC (Stampf, 1994) (to unroll CIFs for use by Unix utilities)] are discussed in Chapter 5.3. In addition there are *CIF2SX* by Louis J. Farrugia (http://www.chem.gla.ac.uk/~louis/software/utils/), to convert from CIF to *SHELXL* format, and *DIFRAC* (Flack *et al.*, 1992) to translate many diffractometer output formats to CIF. The program *cif2xml* (Bernstein & Bernstein, 2002) translates from CIF to XML and CML. The PDB

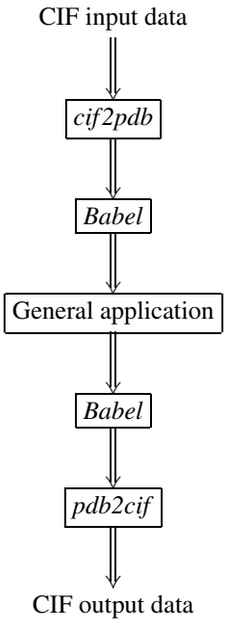


Fig. 5.1.3.2. Example of using filters to make a general application CIF-aware.

provides *CIFTr* by Zukang Feng and John Westbrook (http://sw-tools.pdb.org/apps/CIFTr/) to translate from the extended mmCIF format described in Appendix 3.6.2 to PDB format and *MAXIT* (http://sw-tools.pdb.org/apps/MAXIT/), a more general package that includes conversion capabilities. See also Chapter 5.5 for an extended discussion of the handling of mmCIF in the PDB software environment.

5.1.3.2. Using existing CIF libraries and APIs

Another approach to making an existing application CIF-aware or to design a new CIF-aware application is to make use of one (or more) of the existing CIF libraries and application programming interfaces (APIs). Because the data involved need not be reprocessed, code that uses a library directly is often faster than equivalent code working with filter programs. The code within an application can be tuned to the internal data structures and coding conventions of the application.

The approach to internal design depends on the language, data structures and operating environment of the application. A few years ago, the precise details of language version and operating system would have been major stumbling blocks to conversion. Today, however, almost every platform supports a variation

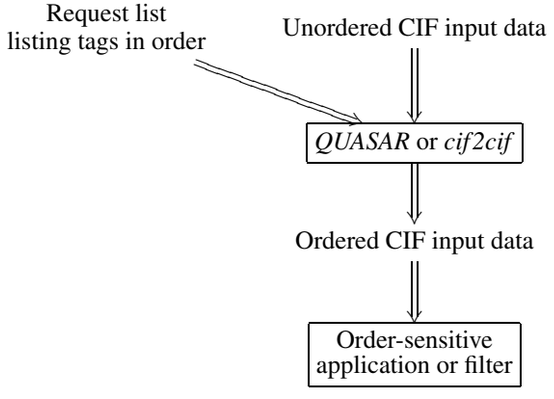


Fig. 5.1.3.3. Using *QUASAR* or *cif2cif* to reorder CIF data for an order-dependent application or filter.

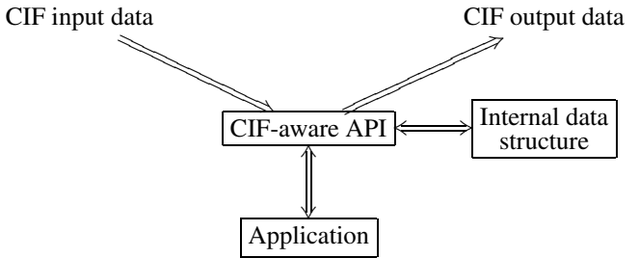


Fig. 5.1.3.4. Typical dataflow of a C-based CIF API.

of the Unix application programming interface and many languages have viable interfaces to C and/or C++. Therefore it is often feasible to consider use of C, C++ or Objective-C libraries, even for Fortran applications. *Star_Base* (Spadaccini & Hall, 1994; Chapter 5.2) is a program for extracting data from STAR Files. It is written in ANSI C and includes the code needed to parse a STAR File. *OOSTAR* (Chang & Bourne, 1998; Chapter 5.2) is an Objective-C package that includes another parser for STAR Files (http://www.sdsc.edu/pb/cif/OOSTAR.html). *CIFLIB* (Westbrook *et al.*, 1997) provides a CIF-specific API. *CIFPARSE* (Tosic & Westbrook, 1998) is another C-based library for CIF. *CBFlib* (Chapter 5.6) is an ANSI C API for both CIF and CBF/imgCIF files. The *CifSieve* package (Hester & Okamura, 1998) provides specialized code generation for retrieval of particular data items in either C or Fortran (see Chapter 5.3 for more details). The package *cciflib* (Keller, 1996) (http://www.ccp4.ac.uk/dist/html/mmcifformat.html) is used by the *CCP4* program suite to support mmCIF in both C and Fortran applications. If an application in Fortran is to be converted with a purely Fortran-based library, the package *CIFtbx* (Hall, 1993; Hall & Bernstein, 1996) is a solution. See Chapter 5.4 for more details.

The common interface provided in C-based applications is for the library to buffer the entire CIF file into an internal data structure (usually a tree), essentially creating a memory-resident database (see Fig. 5.1.3.4). This preload greatly reduces any demands on the application to deal with the order-independence of CIF, at the expense of what can be a very high demand for memory. The problem of excessive memory demand is dealt with in *CBFlib* by keeping large text fields on disk, with only pointers to them in memory. In some libraries, validation of tags against dictionaries is handled by the API. In others it is the responsibility of the application programmer. While the former approach helps to catch errors early, the second, 'lightweight' approach is more popular when fast performance is required.

The most commonly used versions of Fortran do not include dynamic memory management. In order to preload an arbitrary CIF, one needs to use one of the C-based libraries. Alternatively, a pure Fortran application can transfer CIFs being read to a disk-based random access file. *CIFtbx* does this each time it opens a CIF. The user never works directly with the original CIF data set. This provides a clean and simple interface for reading, but slows all read access to CIFs. In Fortran, compromises are often necessary, with critical tables handled in memory rather than on disk, but this may force changes in dimensions and then recompilation when dictionaries or data sets become larger than anticipated.

5.1.3.3. Creating a CIF-aware application from scratch

The primary disadvantage of using an existing CIF library or API in building an application is that there can be a loss of performance or a demand for more resources than may be needed. The common practice followed by most libraries of building and

```
cbf:              datablock            { cbf_failnez (cbf_find_parent (&($$), $1, CBF_ROOT)) }
                ;
cbfstart:                              { $$ = ((void **) context) [1]; }
                ;
datablockstart:   cbfstart             { cbf_failnez (cbf_make_child (&($$), $1, CBF_DATABLOCK, NULL)) }
                | cbf datablockname    { cbf_failnez (cbf_make_child (&($$), $1, CBF_DATABLOCK, $2)) }
                ;
datablock:        datablockstart       { $$ = $1; }
                | assignment           { cbf_failnez (cbf_find_parent (&($$), $1, CBF_DATABLOCK)) }
                | loopassignment       { cbf_failnez (cbf_find_parent (&($$), $1, CBF_DATABLOCK)) }
                ;
category:         datablock categoryname  { cbf_failnez (cbf_make_child (&($$), $1, CBF_CATEGORY, $2)) }
                ;
column:           category columnname  { cbf_failnez (cbf_make_child (&($$), $1, CBF_COLUMN, $2))  }
                | datablock itemname   { cbf_failnez (cbf_make_new_child (&($$), $1, CBF_CATEGORY, NULL))
                                         cbf_failnez (cbf_make_child (&($$), $$, CBF_COLUMN, $2)) }
                ;
assignment:       column value         { $$ = $1;
                                         cbf_failnez (cbf_set_columnrow ($$, 0, $2, 1)) }
                ;
loopstart:        datablock loop       { cbf_failnez (cbf_make_node (&($$), CBF_LINK, NULL, NULL))
                                         cbf_failnez (cbf_set_link ($$, $1)) }
                ;
loopcategory:     loopstart categoryname  { cbf_failnez (cbf_make_child (&($$), $1, CBF_CATEGORY, $2))
                                            cbf_failnez (cbf_set_link ($1, $$))
                                            $$ = $1; }
                | loopcolumn categoryname { cbf_failnez (cbf_find_parent (&($$), $1, CBF_DATABLOCK))
                                            cbf_failnez (cbf_make_child (&($$), $$, CBF_CATEGORY, $2))
                                            cbf_failnez (cbf_set_link ($1, $$))
                                            $$ = $1; }
                ;

loopcolumn:       loopstart itemname    { cbf_failnez (cbf_make_new_child (&($$), $1, CBF_CATEGORY, NULL))
                                          cbf_failnez (cbf_make_child (&($$), $$, CBF_COLUMN, $2))
                                          cbf_failnez (cbf_set_link ($1, $$))
                                          cbf_failnez (cbf_add_link ($1, $$))
                                          $$ = $1; }
                | loopcolumn itemname   { cbf_failnez (cbf_find_parent (&($$), $1, CBF_DATABLOCK))
                                          cbf_failnez (cbf_make_child (&($$), $$, CBF_CATEGORY, NULL))
                                          cbf_failnez (cbf_make_child (&($$), $$, CBF_COLUMN, $2))
                                          cbf_failnez (cbf_set_link ($1, $$))
                                          cbf_failnez (cbf_add_link ($1, $$))
                                          $$ = $1; }
                | loopcategory columnname  { cbf_failnez (cbf_make_child (&($$), $1, CBF_COLUMN, $2))
                                          cbf_failnez (cbf_set_link ($1, $$))
                                          cbf_failnez (cbf_add_link ($1, $$))
                                          $$ = $1; }
                ;
loopassignment:   loopcolumn value      { $$ = $1;
                                          cbf_failnez (cbf_shift_link ($$))
                                          cbf_failnez (cbf_add_columnrow ($$, $2))  }
                | loopassignment value  { $$ = $1;
                                          cbf_failnez (cbf_shift_link ($$))
                                          cbf_failnez (cbf_add_columnrow ($$, $2))  }
                ;
loop:             LOOP
                ;
datablockname:    DATA                 { $$ = $1; }
                ;
categoryname:     CATEGORY             { $$ = $1; }
                ;
columnname:       COLUMN               { $$ = $1; }
                ;
itemname:         ITEM                 { $$ = $1; }
                ;
value:            STRING               { $$ = $1; }
                | WORD                 { $$ = $1; }
                | BINARY               { $$ = $1; }
                ;
```

Fig. 5.1.3.5. Example of *bison* data defining a CIF parser (taken from *CBFlib*).

preloading an internal data structure that holds the entire CIF may not be the optimal choice for a given application. When reading a CIF it is difficult to avoid the need for extra data structures to resolve the issue of CIF order independence. However, when writing data to a CIF, it may be sufficient simply to write the necessary tags and values from the internal data structures of an application, rather than buffering them through a special CIF data structure.

It is tempting to apply the same reasoning to the reading of CIF and create a fixed ordering in which data are to be processed, so that no intermediate data structure will be needed to buffer a CIF.

Unless the application designer can be certain that externally produced CIFs will never be presented to the application, or will be filtered through a reordering filter such as *QUASAR* or *cif2cif*, working with CIFs in an order-dependent mode is a mistake.

Because of the importance of being able to accept CIFs written by any other application, which may have written its data in a totally different order than is expected, it is a good idea to make use of one of the existing libraries or APIs if possible, unless there is some pressing need to do things differently.

If a fresh design is needed, *e.g.* to achieve maximal performance in a time-critical application, it will be necessary to create a CIF parser to translate CIF documents into information in the internal data structures of the application. In doing this, the syntax specification of the CIF language given in Chapter 2.2 should be adhered to precisely. This result is most easily achieved if the code that does the parsing is generated as automatically as possible from the grammar of the language. Current 'industrial' practice in creating parsers is based on use of commonly available tools for lexical scanning of tokens and parsing of grammars based on *lex* (Lesk & Schmidt, 1975) and *yacc* (Johnson, 1975). Two accessible descendants of these programs are *flex* (by V. Paxson *et al.*) and *bison* (by R. Corbett *et al.*). See Fig. 5.1.3.5 for an example of *bison* data in building a CIF parser. Both *flex* and *bison* are available from the GNU project at http://www.gnu.org.

Neither *flex* nor *bison* is used directly by the final application. Each may be used to create code that becomes part of the application. For example, both are used by *CifSieve* to generate the code it produces. There is an important division of labour between *flex* and *bison*; *flex* is used to produce a lexicographic scanner, *i.e.* code that converts a string of characters into a sequence of 'tokens'. In CIF, the important tokens are such things as tags and values and reserved words such as `loop_`. Once tokens have been identified, responsibility passes to the code generated by *bison* to interpret. In practice, because of the complexities of context-sensitive management of white space to separate tokens and the small number of distinct token types, *flex* is not always used to generate the lexicographic scanner for a CIF parser. Instead, a hand-coded lexer might be used.

The parser generated by *bison* uses a token-based grammar and actions to be performed as tokens are recognized. There are two major alternatives to consider in the design: event-driven interaction with the application or building of a complete data structure to hold a representation of the CIF before interaction with the application. The advantage of the event-driven approach is that a full extra data structure does not have to be populated in order to access a few data items. The advantage of building a complete representation of the CIF is that the application does not have to be prepared for tags to appear in an arbitrary order.

5.1.4. Conclusion

Making CIF-aware applications is a demanding, but manageable, task. A software developer has the choice of using external filters, using existing libraries and APIs, or of building CIF infrastructure from scratch. The last choice presents an opportunity to tune the handling of CIFs to the needs of the application, but also presents the risk of creating code that does not conform to CIF specifications. One can never know for certain how a new application may be used in the future. If there is any doubt that an application built from scratch will conform to CIF specifications, prudence dictates that one should use filter programs or well tested libraries and APIs in preference to cutting corners in building an application from scratch.

We are grateful to Frances C. Bernstein for her helpful comments and suggestions.

References

Allen, F. H. (2002). *The Cambridge Structural Database: a quarter of a million crystal structures and rising. Acta Cryst.* B**58**, 380–388.

Allen, F. H., Kennard, O., Motherwell, W. D. S., Town, W. G. & Watson, D. G. (1973). *Cambridge Crystallographic Data Centre. II. Structural Data File. J. Chem. Doc.* **13**, 119–123.

Andrews, N. (1987). *Rich Text Format standard makes transferring text easier. Microsoft Syst. J.* **2**, 63–67.

Berners-Lee, T. (1989). *Information management: a proposal.* Internal Report. Geneva: CERN. http://www.w3.org/History/1989/proposal-msw.html.

Bernstein, F. C. & Bernstein, H. J. (1996). *Translating mmCIF data into PDB entries. Acta Cryst.* A**52** (Suppl.), C-576.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F. Jr, Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *The Protein Data Bank: a computer-based archival file for macromolecular structures. J. Mol. Biol.* **112**, 535–542.

Bernstein, H. J. (1997). *cif2cif – CIF copy program*. Bernstein + Sons, Bellport, NY, USA. Included in http://www.bernstein-plus-sons.com/software/ciftbx.

Bernstein, H. J. & Bernstein, F. C. (2002). *YAXDF and the interaction between CIF and XML. Acta Cryst.* A**58** (Suppl.), C257.

Bernstein, H. J., Bernstein, F. C. & Bourne, P. E. (1998). *CIF applications. VIII. pdb2cif: translating PDB entries into mmCIF format. J. Appl. Cryst.* **31**, 282–295. Software available from http://www.bernstein-plus-sons.com/software/pdb2cif.

Bray, T., Paoli, J. & Sperberg-McQueen, C. (1998). *Extensible Markup Language (XML)*. W3C recommendation 10-February-1998. http://www.w3.org/TR/1998/REC-xml-19980210.

Cambridge Structural Database (1978). *Cambridge Crystallographic Database User Manual*. Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge, England.

Chang, W. & Bourne, P. E. (1998). *CIF applications. IX. A new approach for representing and manipulating STAR files. J. Appl. Cryst.* **31**, 505–509.

Diamond, R. (1971). *A real-space refinement procedure for proteins. Acta Cryst.* A**27**, 436–452.

Dubuisson, O. (2000). *ASN.1 – communication between heterogeneous systems*. San Francisco, CA: Morgan Kaufmann. (Translated from the French by P. Fouquart.)

Flack, H. D., Blanc, E. & Schwarzenbach, D. (1992). *DIFRAC, single-crystal diffractometer output-conversion software. J. Appl. Cryst.* **25**, 455–459.

Hall, S. R. (1993). *CIF applications. IV. CIFtbx: a tool box for manipulating CIFs. J. Appl. Cryst.* **26**, 482–494.

Hall, S. R. & Bernstein, H. J. (1996). *CIF applications. V. CIFtbx2: extended tool box for manipulating CIFs. J. Appl. Cryst.* **29**, 598–603.

Hall, S. R. & Sievers, R. (1993). *CIF applications. I. QUASAR: for extracting data from a CIF. J. Appl. Cryst.* **26**, 469–473.

Hammersley, A. P. (1997). *FIT2D: an introduction and overview*. ESRF Internal Report ESRF97HA02T. Grenoble: ESRF.

Heller, S. R., Milne, G. W. A. & Feldmann, R. J. (1977). *A computer-based chemical information system. Science*, **195**, 253–259.

Hester, J. R. & Okamura, F. P. (1998). *CIF applications. X. Automatic construction of CIF input functions: CifSieve. J. Appl. Cryst.* **31**, 965–968.

ISO (1986). ISO 8879. *Information processing – Text and office systems – Standard Generalized Markup Language (SGML).* Geneva: International Organization for Standardization.

ISO (2002). ISO/IEC 8824-1. *Abstract Syntax Notation One (ASN.1). Specification of basic notation.* Geneva: International Organization for Standardization.

Johnson, S. C. (1975). *YACC: Yet Another Compiler-Compiler*. Bell Laboratories Computing Science Technical Report No. 32. Bell Laboratories, Murray Hill, New Jersey, USA. (Also in UNIX Programmer's Manual, Supplementary Documents, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version, March 1984.)

Keller, P. A. (1996). *A mmCIF toolbox for CCP4 applications*. *Acta Cryst.* A**52** (Suppl.), C-576.

Klosowski, P., Koennecke, M., Tischler, J. Z. & Osborn, R. (1997). *NeXus: a common format for the exchange of neutron and synchrotron data. Physica B Condens. Matter*, **B241–243**, 151–153.

Knuth, D. E. (1986). *The TEXbook. Computers and typesetting*, Vol. A. Reading, MA: Addison-Wesley.

Lesk, M. E. & Schmidt, E. (1975). *Lex – a lexical analyzer generator*. Bell Laboratories Computing Science Technical Report No. 39. Bell Laboratories, Murray Hill, New Jersey, USA. (Also in UNIX Programmer's Manual, Supplementary Documents, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version, March 1984.)

Lykos, P. (1975). Editor. *Computer networking and chemistry*. ACS Symposium Series, Vol. 19. Washington DC: American Chemical Society.

Murray-Rust, P. & Rzepa, H. (1999). *Chemical markup, XML and the WWW, Part I: Basic principles. J. Chem. Inf. Comput. Sci.* **39**, 928–942.

NCSA (1993). *NCSA HDF: specification and developer's guide*. Version 3.2. University of Illinois at Urbana-Champaign, USA.

Ohkawa, H., Ostell, J. & Bryant, S. (1995). *MMDB: an ASN.1 specification for macromolecular structure.* In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, Cambridge, England, 16–19 July 1995*, pp. 259–267. Menlo Park, CA: American Association for Artificial Intelligence.

PDB (1974). *PDB Newsletter 1*. Brookhaven National Laboratory, USA.

PDB (1976*a*). *PDB Newsletter 2*. Brookhaven National Laboratory, USA.

PDB (1976*b*). *PDB Newsletter 3*. Brookhaven National Laboratory, USA.

PDB (1977). *PDB Newsletter 4*. Brookhaven National Laboratory, USA.

Spadaccini, N. & Hall, S. R. (1994). *Star_Base: accessing STAR File data. J. Chem. Inf. Comput. Sci.* **34**, 509–516.

Stampf, D. R. (1994). *ZINC – galvanizing CIF to work with UNIX*. Manual. Protein Data Bank, Brookhaven National Laboratory, USA.

Tosic, O. & Westbrook, J. D. (1998). *CIFPARSE: A library of access tools for mmCIF. Reference guide*. Version 3.1. Nucleic Acid Database Project, Department of Chemistry and Chemical Biology, Rutgers, The State University of New Jersey, USA. http://sw-tools.pdb.org/apps/CIFPARSE/cifparse/cifparse.html.

Walters, P. & Stahl, M. (1994). *BABEL reference manual*. Version 1.06. Dolata Research Group, Department of Chemistry, University of Arizona, USA.

Westbrook, J. D., Hsieh, S.-H. & Fitzgerald, P. M. D. (1997). *CIF applications. VI. CIFLIB: an application program interface to CIF dictionaries and data files. J. Appl. Cryst.* **30**, 79–83.

5.2. STAR File utilities

BY N. SPADACCINI, S. R. HALL AND B. MCMAHON

5.2.1. Introduction

The STAR File, described in Chapter 2.1, has a simple format intended to allow the flexible and extensible representation of data without regard to specific data models. In crystallography and related disciplines, the restricted format chosen for the Crystallographic Information File (CIF, Chapter 2.2) and Crystallographic Binary File (CBF, Chapter 2.3) lends itself to rather flat data models. In particular, the relationships between data items enforced through DDL2 dictionaries in applications such as mmCIF (Chapter 3.6) are essentially equivalent to the data structures and relationships of a relational database. Of course, properly normalized relational tables can represent a hierarchy of structure, although this may not be an efficient representation.

There are other applications, such as the molecular information file (MIF, Chapter 2.4), that make use of additional features of the STAR File, such as multiple-level loop structures, global variable scoping and data-instance encapsulation in save frames. These applications may more efficiently represent certain hierarchical or object-oriented data models.

While particular applications require software tools tailored to their specific purposes, it is helpful to have programs or libraries capable of manipulating arbitrary STAR File data, relying solely on the syntax rules and format of the STAR File and taking no account of the semantic content of the included data.

In this chapter, the stand-alone program *Star_Base* is described in detail. This program uses a local query language to demonstrate the ability to retrieve or re-order data with their associated context. There is also a brief review of *Star.vim* and *StarMarkUp*, applications for editing and browsing STAR Files. The chapter concludes by reviewing a number of object classes and libraries for a variety of STAR and generalized CIF applications: prototypical approaches *OOSTAR* and *CIF++*, *CIFOBJ* and *starlib* used by major macromolecular data repositories, and the document-object model package *StarDOM*.

5.2.2. Data instances and context

In a STAR File, a data item consists of a *value*, which is a simple ASCII character string, and an associated identifier or *data name* which precedes the value, and is invariably an ASCII character string beginning with an underscore character and not including any white-space character, such as `_date` or `_chemical_formula_sum`. (The detailed and formal syntax rules for STAR Files are given in Chapter 2.1.)

5.2.2.1. Single and multiple values

A data item may have a single value, in which case the data name may immediately precede the data value, separated only by white space, *e.g.*

```
_chapter_title  'STAR File utilities'
```

Alternatively, a data item may occur multiple times, in a vector or a list. In such a case, the data identifiers appear in a loop header and the values follow in the order of presentation in the loop header. For the simple example of a tabular array, the loop header plays the role of column header, *e.g.*

```
loop_
   _chapter_number
   _chapter_title
       5.2  'STAR File utilities'
       5.3  'Syntactic utilities for CIF'
```

Here the instances of the data item identified by the data name `_chapter_number` have two values, `5.2` and `5.3`. Likewise the instances of the data item identified by `_chapter_title` have two values.

Note an important point: the example has been chosen to suggest to the reader a tabular relationship between the two data items, and in many STAR File applications such a relationship is intended and perhaps formalized through an external dictionary defining the relationships between these data names. However, *the existence of such a relationship is not mandated by the STAR File syntax*. It is legitimate for a generic STAR application to extract a single data item from such an aggregated loop without making any supposition about its relationship with other data items in the same loop. (It should be emphasized that in practice such physical juxtaposition of data items will almost invariably represent a real relationship, and that most application-specific programming will depend on this fact; but it is not an essential component of STAR in its most abstract form.)

It is also axiomatic that the ordering of the multiple values within a list structure has no intrinsic significance in the STAR paradigm. (Again, specific applications may override this by enforcing an ordering, but this is not fundamental to STAR.)

5.2.2.2. Loop packets and context within lists

Where multiple data names are declared in a loop header, STAR does however enforce the notion of a 'loop packet'. The loop packet is the data structure including all individual data values at a particular iteration through the loop. Hence, in the simple example above, `5.2` and `STAR File utilities` comprise the tuple of values in a single loop packet. For the single level of loop considered so far, the loop packet plays the role of a table row.

For nested loops, the situation is more complex. Consider Fig. 5.2.2.1, which is an example of quantum chemistry basis sets for hydrogen and lithium. (The examples in this chapter are derived from various test applications, and do not represent specific adopted exchange protocols in the selected subject areas.) For each element, a list of basis sets is presented, each containing a set of parameters and a table of functional values. At the outermost level of looping in this example, a loop packet comprises all the data associated with an individual atom type, for example hydrogen. At the next inner level of looping, a loop packet corresponds to an individual basis set (including its embedded table of

Affiliations: N. SPADACCINI, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

```
data_Gaussian

loop_
     _basis_set_atomic_name
     _basis_set_atomic_symbol
     _basis_set_atomic_number
     _basis_set_atomic_mass
    loop_
     _basis_set_contraction_scheme
     _basis_set_funct_per_contraction
     _basis_set_primary_reference
     _basis_set_source_exponent
     _basis_set_source_coefficient
     _basis_set_atomic_energy
        loop_
     _basis_set_function_exponent
     _basis_set_function_coefficient

      hydrogen    H    1    1.0079
#     --------
 (2)->[2]   1:    PKC1.1.1   R44 .       -0.485813
      1.3324838E+01    1.0
      2.0152720E-01    1.0 stop_
 (2)->[2]   1:    PKC1.2.1   R33 .       -0.485813
      1.3326990E+01    1.0
      2.0154600E-01    1.0 stop_
 (2)->[1]   2     PKC1.14.1  R24 R24     -0.485813
      1.3324800E-01    2.7440850E-01
      2.0152870E-01    8.2122540E-01 stop_
 (3)->[2]   2:1   PKC1.23.1  R75 R75     -0.496979
      4.5018000E+00    1.5628500E-01
      6.8144400E-01    9.0469100E-01
      1.5139800E-01    1.0000000E+01 stop_ stop_

     lithium    Li   3     6.94
#    -------
 (4)->[4]   1:    PKC3.1.1   R44 .       -7.376895
       3.4856175E+01     1.0
       5.1764114E+00     1.0
       1.0514394E+00     1.0
       4.7192775E-02     1.0 stop_
 (9,4)->[3,2]  7:2:1,3:1
                  PKC3.9.1   R2  R98     -7.431735
       921.271    0.001367       138.730    0.010425
       31.9415    0.049859       9.35329    0.160701
       3.15789    0.344604       1.15685    0.425197
       0.44462    0.169468       0.44462   -0.222311
       0.076663   1.116477       0.028643   1.0
       1.488      0.038770       0.2667     0.236257
       0.07201    0.830448       0.02370    1.0     stop_
 (4,3)->[3,2]   4:2:1,2:1
                  PKC3.30.1  R77 R77     -7.419509
       1.09353E+02    1.90277E-02 1.64228E+01    1.30276E-01
       3.59415E+00    4.39082E-01 9.05297E-01    5.57314E-01

       5.40205E-01   -2.63127E-01 1.02255E-01    1.14339E+00
       2.85645E-02    1.00000E+00

       5.40205E-01    1.61546E-01 1.02255E-01    9.15663E-01
       2.85645E-02    1.00000E+00  stop_ stop_
```

Fig. 5.2.2.1. Example quantum chemistry basis set functions in STAR File format.

coefficients). At the innermost loop level, a loop packet is simply a row within a table of exponents and coefficients of the basis set function.

If one were to treat this example file as a database of indeterminate structure and query the values associated with one of the data names, for example, `_basis_set_function_exponent`, one would retrieve a series of strings `1.3324838E+01`, `2.0152720E-01` *etc.* However, the value strings in themselves are insufficient to allow the reconstitution of any data structure in the file. One also needs an expression of the levels within the nested loop structure at which the values were located, and an indication that they were associated with different packets of information at those various levels. This additional information about the context of each value is sufficient to determine its position within the data structure

```
data_Gaussian
loop_
    loop_
        loop_
     _basis_set_function_exponent
        stop_
    stop_

        1.3324838E+01 2.0152720E-01 stop_
        1.3326990E+01 2.0154600E-01 stop_
        1.3324800E-01 2.0152870E-01 stop_
        4.5018000E+00 6.8144400E-01 1.5139800E-01 stop_
    stop_

        3.4856175E+01 5.1764114E+00 1.0514394E+00
        4.7192775E-02 stop_

        921.271 138.730 31.9415 9.35329 3.15789 1.15685
        0.44462 0.44462 0.076663 0.028643 1.488 0.2667
        0.07201 0.02370 stop_

        1.09353E+02 1.64228E+01 3.59415E+00 9.05297E-01
        5.40205E-01 1.02255E-01 2.85645E-02 5.40205E-01
        1.02255E-01 2.85645E-02 stop_
    stop_
```

Fig. 5.2.2.2. Retrieval from the example file in Fig. 5.2.2.1 of the value of `_basis_set_function_exponent` with associated context.

without any other *a priori* information regarding the data model. The context is most easily expressed by listing the output values in STAR File format.

Fig. 5.2.2.2 is an output listing of the requested values for this example, where the context is expressed as the innermost of three nested loop levels and distinct packets at this level are indicated. It will be seen also that by tracing the disposition of `stop_` words the embedding within higher-level loop packets can also be inferred.

5.2.2.3. Context in data sets

Another indicator of context in the previous example is the data-block header, which was reproduced in the output of Fig. 5.2.2.2.

The STAR File allows data instances in three types of location: in a data block, in a save frame or in a global block.

The usual way to partition a STAR File is by data blocks; each such block represents a data set in which a data name (associated with a single or multiple values) may be declared once only.

Data blocks may include save frames. A save frame is an encapsulated subsidiary data set, effectively insulated from the contents of the surrounding data block, in which data items may occur that have the same names as items in the parent data block. Indeed, 'parent' is potentially a misleading term, since no relationship is implied between the data within a save frame and those in the data block in which the save frame occurs. A reference to a save frame may, however, occur as a data *value* within the data block where the save frame is specified. Recall from Section 2.1.3.6 that save frames within a data block are uniquely identified by the *frame-code* header.

Global blocks may also occur in a STAR File, preceding or interspersed between data blocks. For each data item defined within a global block, that definition is inherited by each succeeding data block that does not contain an internal definition of a data item with the same name. If there is a definition of a data item with the same name within a data block, that internal definition overrides the global definition within that data block. The situation is then re-evaluated in the next data block. If that data block does not contain an internal definition, the global definition holds.

```
data_reaction

save_methyl
  loop_
     _atom_identity_node
     _atom_identity_symbol      1  C  2  C
  loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  1  3
save_

save_ethyl
  loop_
     _atom_identity_node
     _atom_identity_symbol      1  C  2  C  3  C
  loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  1  3  2  3
save_

save_R1
  loop_
     _variable_alternative_number
     _variable_identifier_symbol
     _variable_node    1  $methyl  1   2  $ethyl  1
save_

save_carboxylic_acid
  loop_
     _atom_identity_node
     _atom_identity_symbol      1 $R1  2  C  3  O  4
O
  loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  2  0  3  0  4  1
save_

loop_
     _reaction_component_number
     _reaction_component_symbol
     _reaction_component_type
            1  $carboxylic_acid  reactant
```

Fig. 5.2.2.3. Example STAR data structure where save frames encapsulate related data sets. See text for details.

The scope of data values is well defined (see Section 2.1.3.9). Only data expressed in a global block have values that are inherited in later portions of the STAR File. Data values in data blocks or save frames are restricted in scope to the current data block or save frame, respectively.

A consequence of these rules of scope and encapsulation is that a full description of the context of a STAR data value must also reflect any values carried through as global data or by de-referencing associated save frames. The results are not always intuitive.

Consider Fig. 5.2.2.3, which represents a partial description of a chemical reaction where one of the reactants is expressed as a generic structure described by the save frame `save_R1`. However, the generic structure in this case is restricted to a small number of alkyl groups, each described in its own save frame. Setting aside this prior knowledge, we see that a request for `_atom_identity_symbol` must return not only the data values in their embedded save frames, but also the save frames in their entirety and the higher-order data values that reference the matching save frames. It is only in this way that we can guarantee that the value can be used by any application. Fig. 5.2.2.4 demonstrates the full context of the returned requested data values.

Notice that the requested item occurs (among other places) in the save frame `save_carboxylic_acid` and this instance of the

```
data_reaction

save_methyl
loop_
     _atom_identity_node
     _atom_identity_symbol    1 C 2 C
loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  1 3
save_

save_ethyl
loop_
     _atom_identity_node
     _atom_identity_symbol    1 C 2 C 3 C
loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  1 3 2 3
save_

save_R1
loop_
     _variable_alternative_number
     _variable_identifier_symbol
     _variable_node    1  $methyl  1  2  $ethyl  1
save_

save_carboxylic_acid
loop_
     _atom_identity_symbol $R1 C O O
save_

loop_
     _reaction_component_symbol $carboxylic_acid
```

Fig. 5.2.2.4. Context for the requested values of `_atom_identity_symbol` in the preceding example. See text for details.

item is presented solely in the context of the save header and closure strings (it is shown in italics in Fig. 5.2.2.4). However, one of the values extracted from this location is the save-frame reference pointer $R1 that identifies `save_R1`, and the complete contents of this save frame are presented (because the data structure represented by the save frame *is* itself one of the values of the requested data item). Further de-referencing of the save-frame pointers within `save_R1` results in the extraction also of the complete save frames `save_methyl` and `save_ethyl`. In this example it is coincidental that there are instances of the requested data item (`_atom_identity_symbol`) within these returned save frames as well.

Notice, however, that establishing the full context of the returned data demands also that data values referencing the `save_carboxylic_acid` frame be presented. In this example, the value of `_reaction_component_symbol` at the outermost level of the data block is returned, a result that may at first seem surprising. It is only in this way that one can be sure that an arbitrary application will have access to the full semantic information carried by the data item.

Data values declared in global blocks should be presented in the same spirit of supplying the complete context in which the value was instantiated, and not simply the value in isolation. For example, given the trivial STAR File

```
global_
     _example   foo
data_1
data_2
     _example   bar
```

a request for `_example` should return the identical file, *not* the interpolated result

```
data_1
     _example   foo
data_2
     _example   bar
```

despite the latter's equivalence purely in terms of the non-contextual values returned.

5.2.3. *Star_Base*: a general-purpose data extractor for STAR Files

The stand-alone application *Star_Base* (Spadaccini & Hall, 1994) provides a facility for performing database-style queries on arbitrary STAR Files. It is generic in nature and makes no assumptions about the nature or organization of the data in a STAR File. It may indeed be used as an application-specific database tool if the user has prior knowledge of the relationships between included data items. However, by faithfully returning context as well as value in the way outlined in Section 5.2.2, it can be applied to any STAR File even without such prior knowledge.

5.2.3.1. Program features

Star_Base is a fully functional STAR File parser and may be used to test the syntactic validity of an input STAR File. It may be used to write an input STAR File directly to the output stream, while validating the structural integrity as the contents are parsed. The input format and comments are discarded on output.

Given a valid input file, *Star_Base* guarantees to write output in fully compliant STAR format.

If a data name is supplied as a request item, *Star_Base* will return the single or multiple values associated with that data name and their associated context according to the principles of Section 5.2.2, *i.e.* all loop structures, data-block headers and global headers will be returned, and save frames will be expanded as required to accommodate de-referencing of frame codes as returned values.

Where multiple data items are requested, *Star_Base* will write their occurrences to its output stream in the order they were requested, *not* in the order of appearance in the input file. This may disturb data relationships that are implicit in the ordering or association of values in the input file, but it is the responsibility of the user to track and retain such associations where they are an essential part of an application-level data model. As emphasized before, a generic STAR tool will make no assumptions about data models and will simply return values and contexts as requested.

To illustrate the effect of this, consider a request for the following data items from the example file of Fig. 5.2.2.1:

```
_basis_set_atomic_name
_basis_set_atomic_symbol
_basis_set_contraction_scheme
```

Star_Base will return the following result:

```
data_Gaussian
loop_
     _basis_set_atomic_name
     _basis_set_atomic_symbol
    loop_
     _basis_set_contraction_scheme
    stop_

    hydrogen H
        (2)->[2]  (2)->[2]  (2)->[1]  (3)->[2]  stop_

    lithium Li
        (4)->[4]  (9,4)->[3,2]  (4,3)->[3,2]  stop_
```

However, if the same items are requested in a different order,

```
_basis_set_atomic_name
_basis_set_contraction_scheme
_basis_set_atomic_symbol
```

the result is structured differently:

```
data_Gaussian
loop_
     _basis_set_atomic_name
    loop_
     _basis_set_contraction_scheme
    stop_
     _basis_set_atomic_symbol

    hydrogen
        (2)->[2]  (2)->[2]  (2)->[1]  (3)->[2]  stop_
    H
    lithium
        (4)->[4]  (9,4)->[3,2]  (4,3)->[3,2]  stop_
    Li
```

In the examples so far, one or more data items have been requested by name. *Star_Base* extends the type of requests that can be made through its own query language. This gives it much of the power of a database query language such as SQL. Three types of query are supported, known as *data*, *conditional* and *branching* requests.

A *data request* is a straightforward generalization of the request by data name. Individual data items may be requested by name, as may individual data blocks or save frames. Wild carding is permitted to generalize the requests. More details are given in Section 5.2.3.2.

A *conditional request* involves one or more conditions; only data items satisfying the conditions are returned. More details are given in Section 5.2.3.3.

A *branching request* applies similar conditions to establish the context in which matching data items occur within the file, but may also apply scoping rules to select among the available contexts. Only data items matching both the conditions imposed on their values *and* the requested scope are returned. It is the existence of such branching conditions that gives *Star_Base* the ability to select data matching the specific requirements of overlying data models. Again, however, it is emphasized that the program itself operates without any semantic awareness of the significance of the data that is implied within the overlaid data model. More details on branching requests are given in Section 5.2.3.4.

5.2.3.2. The *Star_Base* data request

A *data request* is the simplest type of query used to extract single items from a file. It may be formed from any of the following string types:

(i) a **name** string, *e.g.* `_atom_identity_symbol`;

(ii) a **block** string, *e.g.* `data_Gaussian`;

(iii) a **frame** string, *e.g.* `save_methyl`.

In accordance with the principles set out earlier in this chapter, data requests satisfy the following rules:

(i) Requested data items are returned with their associated context (*i.e.* including the headers of any containing data blocks, save frames and loop structures).

(ii) A request for a data block returns all preceding global blocks (since the data block will contain by inheritance all values in the global blocks).

(iii) A request for a save frame also returns the header of the data block encompassing the save frame. All frame-pointer codes are resolved so that if a requested save frame contains pointer codes to other save frames, these are also returned.

(iv) A request for `global_` returns all global blocks, together with all data-block headers in their scope.

(v) The request need not be specified explicitly. Two wild-card characters are permitted. An asterisk (*) represents *any sequence*

Table 5.2.3.1. *Permitted constructions for a Star_Base conditional request*

<data request>
<data request> <operator> <text string>
<conditional request> & <conditional request>
<conditional request> | <conditional request>
!<conditional request>

of characters and a question mark (?) represents *any single* character.

(vi) A request for looped data returns the items in the order requested, making any necessary adjustments to the structuring of nested loops to preserve the original context.

(vii) A request for data within a save frame returns those items plus the associated context.

(viii) If a requested data item includes a save-frame pointer as a value, the referenced save frame is returned intact. All other pointers contained within the returned data are resolved.

(ix) A request for a data item in a global data block will also return the data-block headers within the scope of the global block.

(x) The scope of a data request is *the entire input file*. Control of the search scope is only possible within branching requests.

5.2.3.3. The *Star_Base* conditional request

While a data request allows retrieval of data items according to *name*, conditional requests allow retrieval of data items by *value*. The general form of a conditional request may be characterized as <data request><operator><text string>, where <data request> is any data request as defined in the preceding section, <operator> is any of the test operators defined below, and <text string> is a string pattern against which values of data items retrieved by the data request are matched according to the operator specified.

Conditional requests may be combined by set operators &, | and ! to provide logical AND, OR and NOT tests. Table 5.2.3.1 lists the allowed constructions for a conditional request. A bare data request is considered a degenerate case of a conditional request.

The construction <conditional request> & <conditional request> allows for the *conjunction* of conditionals. All data are returned (including context) from the intersection of sets of data that individually satisfy the conditions to be a non-empty set.

It is important to note that the conjunction of conditionals based on different data names is the empty set.

The construction <conditional request> | <conditional request> allows for the *disjunction* of conditionals. All data are returned (including context) from the union of sets of data that individually satisfy the conditions to be a non-empty set.

The construction !<conditional request> allows for the *negation* or *complement* of conditionals. All data are returned (including context) from the universal set of data that do not satisfy the conditions of the conditional request. The universal set is defined as the input file.

Table 5.2.3.2 lists the permitted value-matching operators when a retrieved data value is compared with a target text string in the basic test <data request><operator><text string> described above. (If the <text string> contains white-space characters, it must be quoted with matching single or double quotes. The test is performed on the value of the text string, *i.e.* the complete text string including white-space characters but omitting the surrounding quote characters.)

Two classes of operators are defined. *Text operators* may be used to test for string equality, substring containment or greater and lesser values (where the 'greater' and 'lesser' values for text strings are based on the ASCII character set ordering sequence).

Table 5.2.3.2. *Value-matching operators in Star_Base conditional requests*

Requests are of the form <data request> <operator> <text string>. The second column describes the relationship that data identified by the <data request> must satisfy against the <text string> in order to be returned as part of the result set.

Operator	Relationship
Text comparison operators:	
∼=	Is identically equal to
?=	Includes as a substring
∼<	Is less than (in ASCII order)
∼>	Is greater than (in ASCII order)
∼!=	Is not identically equal to
?!=	Does not include as a substring
∼<=	Is not greater than (in ASCII order)
∼>=	Is not less than (in ASCII order)
Numerical comparison operators:	
=	Is equal to
<	Is less than
>	Is greater than
!=	Is not equal to
<=	Is not greater than
>=	Is not less than

These tests are valid for any STAR application. *Numerical operators* permit comparison of the numerical values implied by the returned data-value strings. Recall from Chapter 2.1 that data values in STAR are specified only as character strings. Casting to different types may be performed by specific applications, but is *not* defined for arbitrary STAR applications. Nevertheless, *Star_Base* recognizes that a majority of STAR applications will in fact specify numeric types, and therefore allows for numerical comparisons based on interpretations of certain value strings according to the conventions adopted by CIF for the **numb** data type (Section 2.2.7.4.7.1). Such values may be given as integers, real numbers or in scientific notation.

5.2.3.4. The *Star_Base* branching request

Both conditional and data requests will retrieve matching data items wherever they may be found in the input file; the scope of the query in both cases is the entire file.

The top-level query type supported by *Star_Base*, the *branching request*, allows selection of sub-requests based on the results of prior tests, and also allows the narrowing or expansion of the scope of a request. The effect is to permit extensive control over the selection of data matching complex conditions. It is this which gives *Star_Base* the power of a database query language.

Note again that the user will in general need prior knowledge of the arrangement of data items within a STAR File in order to compose meaningful requests; *Star_Base* is agnostic about the organization and structure of the contents of a data file and will simply return exactly those data items and their context that match the specified conditions.

A branching request takes the following form:

if_ <condition> <branch_request>
  [**else_** <branch_request>]
  [**unknown_** <branch_request>]
endif_

The <condition> has exactly the same form as a conditional request, but does not return data to the calling process. It returns only a logical value that is used to determine which branch to evaluate. This logical return value may be TRUE if the condition is satisfied, UNKNOWN if the condition is not satisfied because there was no occurrence of a requested data name within the current

scope of the query, or FALSE if the condition is not met otherwise.

The branching request must have a condition and a branch request that is made if the condition returns TRUE. The other possible branches are optional. The **unknown_** branch, if present, is executed when the condition returns a value of UNKNOWN. If there is no **unknown_** branch, then the default truth value is set as FALSE (*i.e.* a return value of UNKNOWN is treated as equivalent to FALSE) and the **else_** branch is executed if present. It is possible to override this behaviour by using the special operator **assume_true_** before a condition. This operator forces the default truth value to TRUE when a condition returns an UNKNOWN value. It is a useful shorthand when the same branch request is applied against a condition that is either TRUE or UNKNOWN. The syntax is
assume_true_ (<condition>).

A <branch request> (that is, the set of actual individual requests within a branching request construct) has three possible forms:

(i) a conditional request,

(ii) a branching request,

(iii) **scope_**<scope setting> <branch request> **endscope_**.

Note carefully the different contexts in which branch requests and nested branching requests may occur.

scope_<scope setting> specifies the range of data to be searched in the input file. The effect of the setting is closed by the **endscope_** statement. The permitted values of <scope setting> are:

(i) **data_item_** restricts the branch request to the data items in the condition,

(ii) **loop_packet_** restricts the branch request to the contents of the loop packet in which data match the condition,

(iii) **loop_structure_** restricts the branch request to the loop structure in which data match the condition,

(iv) **save_frame_** restricts the branch request to the contents of the save frame in which data match the condition,

(v) **data_block_** restricts the branch request to the contents of the data block in which data match the condition,

(vi) **file_** specifies that the branch request applies to the contents of the file containing data matching the condition (the default setting).

The default scope is invoked when a **scope_**<scope setting> is *not* specified; in such a case the scope of the branch request is the same as that of the condition.

Fig. 5.2.3.1 demonstrates the construction of a branching request that restricts the scope of the query. The two requests in this figure are applied to the STAR File example of Fig. 5.2.2.1. In Figure 5.2.3.1(*a*), the query is targeted to retrieve all data items in a loop packet where the value of `_basis_set_contraction_scheme` includes the substring `(3)`, provided that the value of `_basis_set_atomic_name` identically matches the string value `hydrogen` in the next outer nested loop packet. The data relevant to the contraction scheme labelled `(3)->[2]` are returned. Note how the wildcard data request `_*` retrieves the data items from the next outer loop structure in which the requested data lie.

In the example of Fig. 5.2.3.1(*c*), the request for an unknown data name cannot be matched within the input file, and the **unknown_** branch of the request is executed. In this case, the secondary request is more specific (only data names including the substring `contraction` are matched) and hence only a few items from the second-level loop are returned. Scopes can be expanded or contracted. If the example of Fig. 5.2.3.1(*a*) were to be modified by replacing the innermost `scope_loop_packet_` declaration with `scope_loop_structure_`, the query would proceed by testing for the existence of a contraction scheme value including the string `(3)` in the loop packet relevant to the hydrogen results (as before). Finding that this condition was satisfied, the result returned would be all data names in

```
if_ _basis_set_atomic_name  = hydrogen
  scope_loop_packet_
  if_ _basis_set_contraction_scheme ?= (3)
    scope_loop_packet_
     _*
    endscope_
  endif_
  endscope_
endif_
```

(*a*)

```
data_Gaussian
loop_
     _basis_set_atomic_name
     _basis_set_atomic_symbol
     _basis_set_atomic_number
     _basis_set_atomic_mass
    loop_
     _basis_set_contraction_scheme
     _basis_set_funct_per_contraction
     _basis_set_primary_reference
     _basis_set_source_exponent
     _basis_set_source_coefficient
     _basis_set_comments_index
     _basis_set_atomic_energy
        loop_
     _basis_set_function_exponent
     _basis_set_function_coefficient
        stop_
    stop_

    hydrogen H 1 1.0079
        (3)->[2] 2:1 PKC1.23.1 R75 R75 C13,C19 -0.496979
            4.5018000E+00 1.5628500E-01
            6.8144400E-01 9.0469100E-01
            1.5139800E-01 1.0000000E+01
            stop_
        stop_
```

(*b*)

```
if_ _basis_set_atomic_name  = hydrogen
  scope_loop_packet_
  if_ _basis_set_contraction_xxxxxx ?= (3)
    scope_loop_packet_
     _*
    endscope_
  unknown_
     _*contraction*
  endif_
  endscope_
endif_
```

(*c*)

```
data_Gaussian
loop_
    loop_
     _basis_set_contraction_scheme
     _basis_set_funct_per_contraction
    stop_

        (2)->[2] 1:    (2)->[2] 1:
        (2)->[1] 2     (3)->[2] 2:1
        stop_
```

(*d*)

Fig. 5.2.3.1. Examples of branching requests and the results returned by *Star_Base* from the example file of Fig. 5.2.2.1. (*a*) A query designed to extract a data structure relevant to one contraction scheme and one atom type. (*b*) The results of that request. (*c*) A similar request, but with a branch followed when the condition cannot be matched against a requested data item in the current scope, and (*d*) the resulting output. See text for discussion.

the encompassing loop *structure – i.e.* in this particular example, the complete loop contents would be returned.

Note that *Star_Base* faithfully returns context even in the processing of complex branching requests. Therefore if, for example, a save-frame pointer is returned as a data value following the processing of a request, the associated save-frame contents will be returned in full so that they are referenced in the returned STAR data structure.

5.2.3.5. Implementation issues

Star_Base is implemented in the C programming language, and exploits Gnu's *flex* and *bison* compiler-compiler system to generate a lexer and parser for the STAR File and a separate lexer and parser for the *Star_Base* query language.

The STAR File parser builds an *in-memory* representation (much like most programming-language compilers) of the file contents, and differs from similar applications that are based on a single pass over a stream (like SAX for XML applications).

While a system like *CIFtbx* retains a block copy of the STAR File in memory, the initial *Star_Base* processing removes all comments and formatting, and stores the meaningful tokens in a binary tree representation. For each STAR File container (global block, data block, save frame, loop or data item) there is a C structure defined. For each of these there are additional structures defined that hold sequences of containers. The nodes of this tree are populated with these structures. Each leaf of the tree is the data item consisting of the data name and its associated value. A binary tree of the global-block sequences is built in reverse order (that is, in an order reverse to that in which they appear in the file), making it simple to identify the global values in scope for a specific data block. It will be recalled that the STAR File semantics require a backward scan through the file to pick up the global blocks in scope.

The binary search algorithm employed is the classic *tsearch* of Knuth (1973), which is part of the standard C libraries. Given modern computer systems, the implementation is extremely fast and efficient. There are no files in existence whose size would test the limits of *Star_Base*.

The use of a binary tree simplifies the process by which a legitimate STAR File is returned as output by *Star_Base* and also how the scope over which the conditionals operate can be controlled by the user. The program stores references to the data nodes of the tree it needs to extract when outputting. Since the location in the original data tree is always stored, the program is easily able to reconstruct the correct structure of the file by walking the tree, identifying the nodes that need to be output in addition to the data.

Star_Base is by default the 'gold standard' for testing other applications for correctness with respect to the syntax and semantics of the STAR File. It can be said that the output of *Star_Base* is not optimal, since it is yet another STAR File and one which is devoid of the original comments and formatting. However *Star_Base* is in essence an API for STAR File applications, rather than a stand-alone program (although it is often used in that way). *Star_Base* was the platform from which BioMagResBank's *starlib* (Section 5.2.6.4) was developed.

5.2.4. Editing STAR Files with *Star.vim*

The *vim* editor supports syntax highlighting for a wide variety of languages through syntax definition rules. The definition rules for STAR Files are simple and small in number. The entire syntax is defined by 19 rules that include the regular expressions for the STAR File keywords, data names, numbers, single- and double-quoted strings, semicolon-delimited text and frame codes. The language for defining a syntax in *vim* is very simple and very powerful. The constructs allow the user to define the syntax precisely enough so that the system does not match patterns within other patterns, unless directed to.

There are three types of syntax items: **keyword**, **match** and **region**.

keyword can only contain keyword characters, no other syntax items. It will only match with a complete word (there are no keyword characters before or after the match). The rule for the STAR syntax is

```
syn keyword strKeyword   global_ save_ loop_ stop_
```

match is a match with a single regular expression (regexp) pattern. The rule for matching on a save-frame code is

```
syn match strFramecode  "$[^ \t]\+"
```

region starts at a match of the 'start' regexp pattern and ends with a match with the 'end' regexp pattern. Any other text can appear in between. There can be several 'start' and 'end' patterns in the one definition. A 'skip' regexp pattern can be used to avoid matching the 'end' pattern. There are a number of character offset parameters that allow the user to redefine the start and end of the matched text given the pattern that matches the regular expression. Quite separately, one can define the region for highlighting, which can be different from the matched text.

The rule that matches a double-quoted string is

```
syn region strString start=+^"+ start=+\s\+"+ms=e
          end=+" +me=e-1 end=+"\t+me=e-1 end=+"$+
           contains=strSpecial " skip=+\\\\\|\\"+
```

In this rule, the beginning of the pattern is a double quote that is either the first character on the line or that has one or more white-space characters before it. The beginning of the matched string (*ms*) is the end of the matched pattern (*e*). That is, the matched string begins at the quote. The end pattern is a double quote followed by a single space, a tab or an end of record. The end of the matched string (*me*) is one character less than the end of the matched pattern (*e*). That is, the trailing character after the closing double quote is not considered part of the matched string. By default the characters between location *ms* and *me* are highlighted. This too can be controlled, and by including $hs = ms + 1$ and $he = me - 1$ the highlighted text would not include the delimiting double quotes.

As these rules are based on regular expressions, there is no possibility of using them to validate the STAR File structure. However, problems in the structure are often identifiable by unexpected or irregular highlighted text [a fact often used in graphical CIF editors to help the user locate visually errors in syntax (see *e.g.* Section 5.3.3.1.4)].

5.2.5. Browser-based viewing with *StarMarkUp*

StarMarkUp is a Tcl/Tk program that takes any STAR File as input and outputs the contents as HTML. The output is a faithful copy of the input, and there is no reformatting or deletion of content.

During transformation, the contents can be cross-referenced against any other STAR File using HTML anchors. This feature is particularly useful when marking up a data file, since the data names contained within can be hyperlinked to their definition in their dictionary. Furthermore, the definitions contained within the dictionary can be hyperlinked to the DDL dictionary. *StarMarkUp* makes no presumptions about the version of DDL employed, the preferred dictionary structure or the specific application

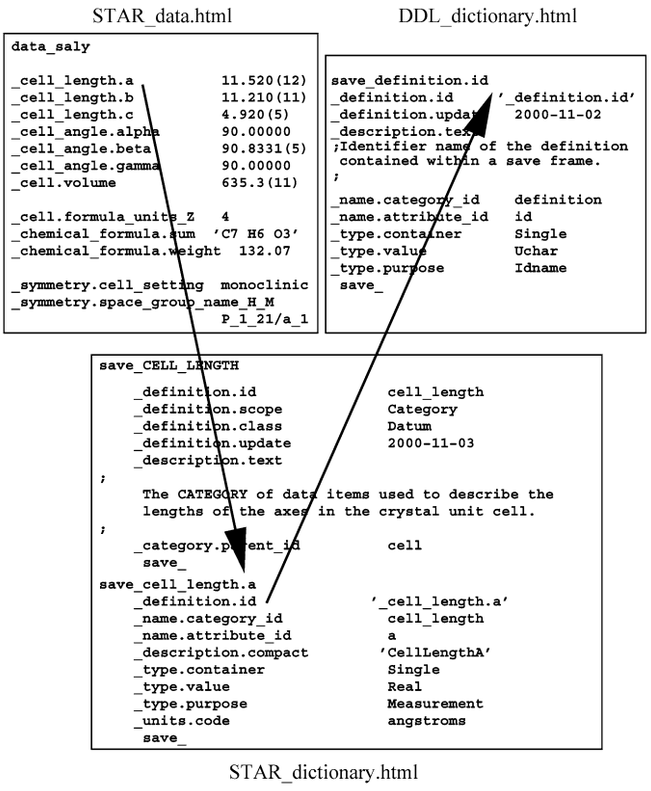


Fig. 5.2.5.1. Hyperlinking markup generated by *StarMarkUp*. The example is based on an extended relational dictionary definition language, StarDDL.

of the STAR File. The program understands only the rules of a legitimate STAR File.

StarMarkUp provides a number of hyperlinking facilities. During markup, it automatically hyperlinks frame-code values to the internal save-frame block to which they point. As part of the same process, *StarMarkUp* inserts anchors in all data and save-frame blocks. It does this in anticipation that there may be the need to hyperlink to these anchors from another STAR File. The most obvious application for this is in the marking up of the DDL dictionary. The list of anchors generated from this process can be passed on to the discipline dictionary during its markup phase (Fig. 5.2.5.1). In this way, the tags used in the discipline dictionary to define the data names can be made to point back to their entry in the DDL dictionary. At the same time that the discipline dictionary is being marked up, a list of its anchors is being generated (each anchor being to a data-item definition). This list is used when marking up an instance of the discipline STAR data file to hyperlink each data name back to its definition in the discipline dictionary.

StarMarkUp comprises a hand-crafted tokenizer employing a character buffer and one-character look-ahead to identify accepted tokens. *StarMarkUp* does not build an internal representation of the file, but functions as a streaming parser. The *GetToken()* function returns a structure consisting of the token type (an enumerated set) and the token value (the lexeme associated with that token). In most cases, the lexeme is marked up and injected into the output stream. If the token is associated with a STAR File block, the parser will recursively call a *MarkUpBlock()* function. A recursive descent parser makes it very easy to treat all STAR File blocks (`global_`, `data_` and `save_`) in identical fashion.

StarMarkUp is implemented in Tcl/Tk for novelty and not because of any particular superior qualities of the language and its API. It is, however, fast and sufficiently flexible, and extensions to the program can be rapidly implemented and tested.

5.2.6. Object-oriented STAR programming

The STAR File syntax is very simple but flexible, and suggests a number of well defined data structures. 'Scalar' values (*i.e.* single text strings identified by specific data names) are obvious, as are 'vectors' (multiple string values associated with a data name in a `loop_` declaration). 'Matrices' are constructed by associating identical-length vector items in the same `loop_` structure. Note that these data structures are not true matrices, but arrays or tables of column-addressable vectors. Column addressing arises from the need to associate each set of values with a distinct data name in the loop header declaration.

Through the nested loop construct in STAR, vector or matrix *elements* are permitted within vectors or matrices. Save frames provide addressable data structures containing one or more scalar, vector or matrix components; and data blocks contain similar data structures with the inclusion of embedded save frames.

This hierarchy of structure allows many other data models to be mapped onto the STAR syntax. For example, STAR does not in itself provide addressable *rows* in matrix structures, but such a concept may be achieved in application-specific ways. For example, the relational database structure of mmCIF and other DDL2 applications ensures that table rows can be identified by specific key values. Loops of multiple values associated with a single data name are similar to *lists* or one-dimensional *arrays* as defined in many computer languages. Simple *associative arrays*, or 'hashes', such as are used to great effect in Perl and Python, can be modelled by designating the first value of a two-item loop structure as the 'key' item, by analogy with the relational database example above (in such an application the key value of course needs to be unique). The hierarchy of nested loops can be used to model certain types of complex structures such as might be defined in a C-language 'struct', for example.

There is a natural possibility of representing particular well defined data structures as 'objects' and handling STAR Files by object-oriented programming techniques. Where applications associate properties of specific data items with external reference descriptions, as in the DDL dictionaries, the dictionary entries can be considered to express types, relationships and even methods associated with the object classes.

However, the versatility of the STAR syntax means that there are many ways of designing STAR objects and their relationships, and it is likely that the most useful efforts will be in designing applications for specific purposes or subject areas. Below we describe a number of prototypes and implementations of object-oriented STAR programming. Some are rather generic in outlook; others are informed by the crystallographic viewpoint embodied in CIF.

5.2.6.1. *OOSTAR*

The *OOSTAR* approach (Chang & Bourne, 1998) developed and tested an Objective-C (Pinson & Richard, 1991) toolset for STAR representation. The developers selected Objective-C over the more common C++ language because of its perceived advantages of (i) loose data typing, (ii) run-time type checking and (iii) message-passing ability. These were all seen to aid flexible software design, an asset for applications built on the very flexible structure of the STAR syntax itself.

Chang & Bourne constructed a set of classes built on the basic objects *item* and *value*. Their model builds upwards through a hierarchy of classes describing relationships between objects. The *ItemAssoc* class contains data members corresponding to the data item itself, its included value or set of values, and the data name that is used to reference the item; the class also has pointers to

previous and next elements to allow iteration. The *StarAssoc* class contains methods for manipulating STAR objects through setting and retrieval of data names and assignment of values. *DataBlock* is the class containing all the lower-order items and associations, and one or more such *DataBlock* classes comprise the *StarFile* class.

The *OOSTAR* approach also provides *Dictionary* and *DictionaryElement* classes so that STAR applications that do describe element attributes in external dictionary files can make use of such information.

Some sample applications were built with the class libraries of the *OOSTAR* toolset, and they are described in Chang & Bourne (1998). They include: simple converters of STAR data files to HTML pages with hyperlinks to associated dictionary entries; a query tool for retrieving the data values for a specific item specified by name within a specific data block; and a query mechanism that retrieves a set of items from a STAR loop structure and represents their values in an array, *i.e.* flattening if necessary any nested loops into a purely tabular presentation. These query tools are different from *Star_Base* primarily in that they do not return the context in which the data are found in the original file – their interpretation depends on a detailed *a priori* understanding of the target file structure.

This approach is rather different from the canonical description of STAR given in this volume, and was never developed into full-blown applications (in part because the developers recognized that the Java language would provide a preferable platform for further development). Nevertheless, it provides interesting ideas for the developer considering building object-oriented applications for STAR Files.

5.2.6.2. *CIF++*

CIF++ was a small library of classes designed by Peter Murray-Rust during the time that DDL was being developed as a language for representing the properties and attributes of STAR data items (*e.g.* Murray-Rust, 1993). The purpose of this project was to demonstrate the design of data classes that represented real-world objects such as molecules and crystal cells. At a time when the structure of CIF was under intense discussion, many of the classes were potentially extensible to STAR features that were a superset of those found in CIF.

While these classes were never developed into fully functional applications, they demonstrated a potentially fruitful approach to model representation and gave rise to the idea of including methods in STAR dictionary definitions, thus allowing STAR applications to dynamically associate algorithmic relationships and operations with data objects by parsing the methods description in the dictionary definitions. This approach is currently being developed into a relational expression language for STAR called dREL (Spadaccini *et al.*, 2000).

The ideas behind *CIF++* were further developed in a class library for molecular representation called *Democritos*, and have informed the representation of molecular and crystal structure in Chemical Markup Language (CML) and in the structured document browser *Jumbo* (Murray-Rust, 1998).

The *CIF++* classes have been refactored into Java using the W3C document object model (W3C, 2004) and other DOM-like models. They are now based on an XML schema which is the abstraction of the formal DDL1 specification (Chapter 2.5).

These are available as part of the *Jumbo* distribution at http://cml.sf.net. The design involves an interface that would allow the C++ classes to be recreated from the Java.

5.2.6.3. *CIFOBJ*

The *CIFOBJ* class library (Schirripa & Westbrook, 1996) was developed to provide an object view of the mmCIF dictionary and to complement the relational *CIFLIB* class library (Westbrook *et al.*, 1997) for handling mmCIF data. As such, it is very much tuned to crystallographic applications and it handles only the subset of STAR features used in CIF dictionaries. This does, however, include some limited handling of save frames, and is therefore a little more complex than applications interested only in CIF data input/output processing.

CIFOBJ has two components. The first builds a persistent store of objects of types item, subcategory, category and dictionary. Each such object is a container for all relevant attributes permitted for that object type. The object store is populated from the mmCIF dictionary by the *CIFOBJ* loader class (using methods provided by *CIFLIB*). This loader class assembles the dictionary objects and passes them to an object-storage manager. The second component of the *CIFOBJ* class library provides the methods necessary for building dictionary objects from the persistent store and for passing attribute strings to procedures concerned with establishing the integrity of data values.

The main reason for discussing this implementation here is to indicate the rapid growth in complexity needed to impose an application-specific object view on even a relatively simple STAR data structure. In generic prototype STAR projects such as *OOSTAR*, half a dozen or so classes and associated methods suffice to represent the highest-level abstract concepts implied in STAR constructions. In *CIFOBJ*, however, dozens of methods are associated with dictionary access. Dictionary-driven validation of an mmCIF numerical data value can involve:

(i) retrieval from the dictionary of the extended type declaration associated with the data item;

(ii) validation of the basic type (*i.e.* that it is indeed numeric) against the primitive data types supported by the dictionary;

(iii) validation of the extended type by regular-expression matching of the string representation of the value against the allowed patterns stored in the dictionary;

(iv) retrieval of any existing range constraints specified in the dictionary and comparison with the data value;

(v) location and evaluation of any associated standard uncertainty;

(vi) identification of the units in which the physical quantity is expressed;

(vii) if necessary, conversion of the units according to the conversion tables stored in the dictionary.

More complexity arises from the relationships between data items expressed through dictionary attributes such as name aliasing, parent–child dependencies and category membership.

Nevertheless, the complexity of these relationships is an indication of the richness of the metadata available through the dictionary approach, and the availability of well defined object representations simplifies the construction of well designed large-scale application frameworks, such as underpin the Protein Data Bank (Berman, Battistuz *et al.*, 2002) and Nucleic Acid Database (Berman, Westbrook *et al.*, 2002).

5.2.6.4. *starlib*

BioMagResBank (BMRB) is a repository for NMR spectroscopy data on proteins, peptides and nucleic acids at the University of Wisconsin – Madison (Ulrich *et al.*, 1989). For some time, NMR data sets have been exchanged within this environment using STAR Files; an NMRStar data dictionary to define the data names used for tagging NMR data is under development

(BioMagResBank, 2004). The *starlib* class library was developed at BMRB for handling NMRStar files, but its initial application to such files independently of the prototype data dictionary means that it is applicable to any STAR File. It does not provide a relational database paradigm (although this is a long-term goal). However, it does provide objects and methods suitable for searching and manipulating STAR data.

Table 5.2.6.1 lists the top-level classes used in *starlib*. *ASTnode* is a formal base class, providing the types and methods that can be used in other derived classes. *StarFileNode* is the root parent of all other objects contained in an in-memory representation of a STAR File; in practice it contains a single *StarListNode*, which is the list of all items contained in the file. *BlockNode* is a class which contains a partition of the STAR File: the class handles both data blocks and global blocks. Data-block names are stored in instances of the *HeadingNode* object, which also holds save-frame identification codes and is therefore useful for accessing named portions of the file.

DataNode is a virtual class representing the types of data objects handled by the library (accessed directly as *DataItemNode*, *DataLoopNode* and *SaveFrameNode*).

Looped data items are handled by a number of objects. *DataLoopNameListNode* is a list of lists of names in a loop. The first list of names is the list of names for the outermost loop, the second list of names is the list of names for the next nesting level and so on. *LoopNameListNode* is a list of tag names representing one single nesting level of a loop's definition. *LoopTableNode* is a table of rows in a *DataLoopNode* (not itemized in Table 5.2.6.1; it is an object representing a list of tag names and their associated values, a particular case of *DataNode*). *starlib* views a loop in a STAR file as a table of values, with each iteration of the loop being a row of the table. Each row of the table can have another table under it (another nesting level), but such tables are the same structure as the outermost one. Thus *LoopTableNode* stores a table at some arbitrary nesting level in the loop. A simple singly nested loop will have only one loop table node, but a multiply nested loop will have a whole tree of loop tables. *LoopRowNode* is a single row of values in a loop.

DataNameNode holds the name of a tag/value pair or a loop tag name. *DataValueNode* is the type that holds a single string value from the STAR file and the delimiter type that is used to quote it.

DataListNode and *SaveFrameListNode* store lists of data within higher-order data objects or save frames, and are internal classes rarely invoked directly by a programmer.

A number of observations may be made regarding this approach. Firstly, the objects can be mapped with reasonable fidelity to the high-level Backus–Naur form representation of STAR (Chapter 2.1). Secondly, it is computationally convenient to abstract common features into parent classes, so that, for example, individual data items, looped data and save frames are represented as child objects of the *DataNode* object, and not themselves as first-generation children of the base class. Thirdly, the handling of nested loops may be achieved in different ways; *starlib* has chosen a particular view that is perhaps well suited to relational data models.

As expected within a programming toolkit, *starlib* offers a large number of methods for retrieving STAR data values, adding new data items, extending or re-ordering list structures, and performing structural transformations of the in-memory data representation. Unlike the stand-alone *Star_Base* application, it does not guarantee that output data will be in a STAR-conformant format; and the programmer is left with the responsibility of validating transformed data at a low level.

Table 5.2.6.1. *Object classes for manipulating STAR data in starlib*

ASTnode	The base class from which all other classes are derived
StarFileNode	The STAR File object
StarListNode	List of items contained in the STAR File
BlockNode	A data or global block
HeadingNode	Labels for major STAR File components
DataNode	General class for data objects
DataLoopNameListNode	List of lists of names in a loop
LoopNameListNode	List of tag names representing one nesting loop level
LoopTableNode	Table of rows in a loop
LoopRowNode	Single row of values in a loop
DataNameNode	A data name
DataValueNode	A single string value
DataListNode	List of data within a higher-order data object
SaveFrameListNode	List of data items allowed in a save frame

Nevertheless, this is a substantial and important library which, as with *CIFOBJ*, has played an important role in the functioning of a major public data repository. Development of the class libraries continues, with a Java version now available.

5.2.6.5. *StarDOM*

A convenience of well designed object representations is that effective transformation between different data representations may be possible. The *StarDOM* package (Linge *et al.*, 1999) demonstrates a transformation from STAR Files to an XML representation, where the tree structure of a STAR File as interpreted in the *starlib* view above is mapped to a document object model (DOM; W3C, 2004). This approach is similar to *Jumbo*, mentioned above in Section 5.2.6.2.

A demonstration of *StarDOM* is the transformation of the complete set of NMR data files at BioMagResBank to XML. The resultant files can then be interrogated using the *XQL* query language (Robie *et al.*, 1998). In this example implementation, the target XML document type definition (DTD) includes a small number of XML elements matching the STAR objects *global* and *data block*, *save frame*, *list*, *data item*, *data name* and *data value*. Particular data names are recorded as values of the <NAME> element. The authors of the *StarDOM* package are considering an extension in which named data items map directly to separate XML elements; the goal is to develop an NMR-specific DTD that is isomorphous to the emerging NMRStar data dictionary.

References

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D. & Zardecki, C. (2002). *The Protein Data Bank. Acta Cryst.* D**58**, 899–907.

Berman, H. M., Westbrook, J., Feng, Z., Iype, L., Schneider, B. & Zardecki, C. (2002). *The Nucleic Acid Database. Acta Cryst.* D**58**, 889–898.

BioMagResBank (2004). *NMR-STAR Dictionary*. Version 3.0. (Under development.) http://www.bmrb.wisc.edu/dictionary/3.0html/.

Chang, W. & Bourne, P. E. (1998). *CIF applications. IX. A new approach for representing and manipulating STAR files. J. Appl. Cryst.* **31**, 505–509.

Knuth, D. E. (1973). *The art of computer programming.* Vol. 3, *Sorting and searching*, pp. 422–447. Reading, MA: Addison-Wesley.

Linge, J. P., Nilges, M. & Ehrlich, L. (1999). *StarDOM: from STAR format to XML. J. Biomol. NMR*, **15**, 169–172.

Murray-Rust, P. (1993). *Analysis of the DDL/dictionary parsing problem. Proc. First Macromolecular Crystallographic Information File (CIF) Tools Workshop*, ch. 12. New York: Columbia University.

Murray-Rust, P. (1998). *The globalization of crystallographic knowledge*. *Acta Cryst.* D**54**, 1065–1070.

Pinson, L. J. & Richard, R. S. (1991). *Objective-C object-oriented programming techniques*. Reading, MA: Addison-Wesley.

Robie, J., Lapp, J. & Schach, D. (1998). *XML Query Language (XQL)*. http://www.w3.org/TandS/QL/QL98/pp/xql.html.

Schirripa, S. & Westbrook, J. D. (1996). *CIFOBJ. A class library of mmCIF access tools.* Version 3.01 reference guide. Technical Report NDB-269. Rutgers University, New Brunswick, New Jersey, USA.

Spadaccini, N. & Hall, S. R. (1994). *Star_Base: accessing STAR File data. J. Chem. Inf. Comput. Sci.* **34**, 509–516.

Spadaccini, N., Hall, S. R. & Castleden, I. R. (2000). *Relational expressions in STAR File dictionaries. J. Chem. Inf. Comput. Sci.* **40**, 1289–1301.

Ulrich, E. L., Markley, J. L. & Kyogoku, Y. (1989). *Creation of a nuclear magnetic resonance data repository and literature database. Protein Seq. Data Anal.* **2**, 23–37.

W3C (2004). *Document Object Model (DOM).* http://www.w3.org/DOM/.

Westbrook, J. D., Hsieh, S.-H. & Fitzgerald, P. M. D. (1997). *CIF applications. VI. CIFLIB: an application program interface to CIF dictionaries and data files. J. Appl. Cryst.* **30**, 79–83.

5.3. Syntactic utilities for CIF

BY B. MCMAHON

5.3.1. Introduction

Since the introduction of the Crystallographic Information File (CIF), the crystallographic community has produced a wide variety of tools and applications to handle CIFs. Many changes have been made to existing programs to input and output CIF data sets, and occasionally changes may have been made to internal crystallographic calculations to provide a better fit to the view of the data expressed by the standard CIF dictionaries. However, for most crystallographers with an involvement in programming, there is an understandable tendency to invest the minimum amount of effort needed to accommodate the new format. Their primary interest is in the understanding and discovery of the underlying physical model of a crystal structure.

This chapter reviews several general-purpose tools that have been developed for CIF to check, edit, extract or manipulate arbitrary data items, with little in the way of crystallographic computation. They are of interest to the end user who wishes to visualize a structure in three dimensions or submit an article to a journal but who does not want to be concerned about the details of CIF. They also include several utilities that are helpful for manipulating the contents of CIFs without the need to write a large and complex program. The programmer with an interest in writing complete and robust CIF applications should look at the comprehensive libraries described in Chapters 5.4 to 5.6.

Many of the programs described in this chapter operate purely at the syntactic level; they require no knowledge of the scientific meaning of the data items being manipulated. Others have some bearing on the *semantics* of the file contents, either explicitly through information about data types and interrelationships carried in external CIF dictionaries, or implicitly through the user's choice and deliberate manipulation of items based on an understanding of what they signify. Nevertheless, most utilities described here are characterized by an ability to handle CIFs of any content and provenance. The best example of a program able to handle arbitrary CIFs at a purely syntactic level is *Star_Base* (Spadaccini & Hall, 1994), described in Chapter 5.2.

It should be noted that not all the programs described here are fully compliant with the specification of Chapter 2.2, and others have implementation restrictions or known bugs. Many, especially the older programs, are no longer actively supported and need to be handled with care. However, they are included here for the record, and because they may provide useful ideas and suggestions to future developers in an area that can still accommodate a wider range of tools for different uses.

5.3.2. Syntax checker

A CIF must conform to a subset of the syntax rules of a general STAR File (Chapter 2.1), but with the additional restrictions and conventions described in Chapter 2.2. The syntax is rather simple and robust subroutines to create CIFs may easily be written by computer programmers. However, the use of ASCII character sets, deliberately expressive data names and simple layout conventions both permit and encourage users to edit the files with general text editors that cannot guarantee to retain syntactic integrity. Consequently, there is a definite use for a simple program that can check whether a file conforms to the specified syntax.

It is worth mentioning that programmable text editors such as *emacs* may be supplied with rules that can check syntax as a file is edited. A simple rule set (known as a *mode file*) has been developed (Winn, 1998) to indicate the different components of a CIF, as a first step towards a syntax-checking *emacs* mode.

The *Star.vim* utility of Section 5.2.4 provides a similar functionality for editing in the *vim* environment, although it is not capable of validation directly; nevertheless, the appearance of unexpected or irregular highlighted text can draw the user's attention to syntactic problems, a feature that is also useful in more extended editors such as *enCIFer* (Section 5.3.3.1).

5.3.2.1. *vcif*

A simple syntax checker for CIF is the program *vcif* (McMahon, 1998), which scans a text file and outputs informative messages about apparent errors. While conservative CIF parsing software will quit upon finding an error, *vcif* will attempt to read to the end of the file and list all clearly distinguished errors. However, its interpretation of errors depends on a close adherence to the CIF syntax specification and makes no assumption about the intended purpose of the character strings it reads. In consequence, a single logical error such as failing to terminate a multiple-line text string may cause the program to report many other apparent errors as it proceeds out of phase through the rest of the file.

5.3.2.1.1. *How to use vcif*

The program may be run under Unix or DOS by typing

```
vcif filename
```

where *filename* is the name of the file to test. If *filename* is given as the hyphen character -, the program will read standard input. Standard input will also be read if no file name is supplied; this allows the program to be used in a pipeline of commands.

A number of options may be supplied to the program to modify its behaviour. Without these options (*i.e.* invoked as above) a brief but informative message is written to the standard output channel for each occurrence of what the program perceives to be a syntax error.

For example, for the incorrect sample file of Fig. 5.3.2.1(*a*), the output is listed in Fig. 5.3.2.1(*b*).

Note that the sequence number of the line in which the error occurs is printed. The summary error message is output on a single line (longer lines have been wrapped and indented in Fig. 5.3.2.1 for legibility). Where the type of error necessarily affects only a single line, the program can recover and correctly identify errors on subsequent lines. Where possible, unexpected character strings are printed to help the user to identify the error. No attempt is made to assign any meaning to the data names or the data values in the

Affiliation: BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

```
# Sample CIF with syntax errors

_date              'Monday 12 April 1999

_cell_length_a     7.514 (3)
_cell_length_b     9.467 (2)

    loop_
     _geom_bond_atom_site_label_1
     _geom_bond_atom_site_label_2
     _geom_bond_distance
                  O1   C2   1.342(4)
                  O1   C5   1.439 (3)

_example_comment
; The purpose of this example is to indicate how vcif
  describes some of the syntax errors it finds.
```

(*a*)

```
ERROR: No data block code before dataname at line 3
ERROR: Single-quoted character string does not
     terminate at line 4
ERROR: Unexpected string ((3)) at line 5
ERROR: Unexpected string ((2)) at line 6
ERROR: Number of loop elements not multiple of
     packetsize at line 15
ERROR: Text field at end of file does not terminate
```

(*b*)

Fig. 5.3.2.1. (*a*) An example CIF with a number of syntax errors and (*b*) the report of the errors produced by *vcif*.

file. Hence the same logical error (the detachment of a standard uncertainty in parentheses from its parent value) is indicated variously as an unexpected text string or as an extraneous loop item, depending on where it occurs in the file. Indeed, in the case of the incorrect number of loop elements, the program makes no attempt to identify which data value or values in the loop might be in error: it simply counts the number of values in a loop and complains when this is not a multiple of the number of data names declared in the loop header.

5.3.2.1.2. *Options to vcif*

A number of options may be supplied as command-line arguments to modify the output from *vcif*.

A more complete account is given of each error on its first occurrence when the program is invoked with the '*-v*' option. The output listing explains in more detail what the breach of syntax is and sometimes suggests how misunderstandings of the file structure result in such breaches (Fig. 5.3.2.2).

Each error message is prefaced by the word 'ERROR' (or occasionally another phrase such as 'WARNING' or 'STAR ERROR'). Three chevrons preface a printout of the beginning of the troublesome line. Then an expanded description of the error is given, prefaced by three asterisks, *on the first occurrence of each distinct error*. In this mode, only the first 20 errors are listed (the assumption is that this mode is best suited to novices, who should identify and correct each error in turn and would not want to be swamped by large numbers of error messages arising from a single error). More errors may be reported by using the '*-e*' command-line option.

The *quiet* option (*vcif -q*) outputs no error messages but instead returns to the calling environment an integer giving the total number of errors found. This option allows scripts or external programs to use *vcif* as a silent test of whether a file has any syntax errors.

```
ERROR: No data block code before dataname at line 3
>>> "_date"
*** A data block MUST begin with a data_something
      declaration.
ERROR: Single-quoted character string does not
      terminate at line 4
>>> "_cell_length_a"
*** The indicated line appears to contain some word
      or words introduced by a single quote, but not
      terminated with a matching single quote.
ERROR: Unexpected string ((3)) at line 5
*** There is an unexpected word or number as
      indicated. This may be because a loop is
      intended but the loop_ keyword has been missed
      out; or a phrase with several words is not
      enclosed in matching delimiting quote marks; or
      a text field (extending over several lines) is
      not properly closed with a final semicolon; or
      a data_ block header has not yet been seen.
ERROR: Unexpected string ((2)) at line 6
>>> "_cell_length_b"
ERROR: Number of loop elements not multiple of
      packetsize at line 15
>>> "_example_comment"
*** A loop_ header defines a list of datanames. The
      values following this header are assigned in
      sequence with the datanames in the header, so
      each packet of information (or row in the table
      of values defined by the loop structure) must
      have the same number of values as there are
      datanames declared in the loop header. Common
      reasons for this error include: omission of a
      value where the associated data are absent
      (insert . or ? as placeholders); numeric values
      where the standard uncertainty (or e.s.d) has
      come adrift from its associated value (e.g.
      10.925 (2)); multi-word phrases or text entries
      that are not properly delimited with quote
      marks or initial semicolons.
ERROR: Text field at end of file does not terminate
>>> ""
*** Is the CIF complete?
```

Fig. 5.3.2.2. Verbose error listing from *vcif* when run with the '*-v*' option on the example of Fig. 5.3.2.1.

A related option, *vcif -b*, counts errors and returns the result as an integer to the calling environment, as in the previous case; but additionally outputs a list of all the data-block codes in the file. While adding nothing to the syntax-checking function of the program, this provides a useful small utility for simply listing data-block names.

Although intended for use with the restricted STAR File syntax permitted for CIF (Chapter 2.2), *vcif* may also be used with the '*-s*' option to check the syntax of CIF dictionary files, which may include save frames. The program does not, however, handle nested loop structures.

The program will flag as an error any line of greater than 80 characters length (the original limit in the CIF version 1.0 specification; see Chapter 2.2), but this behaviour may be overridden with the '*-l*' option. If used, only lines longer than the specified number of characters will be reported and the reports of such lines will be prefaced with the word 'WARNING'. Likewise, the '*-w*' option may be used to override the CIF version 1.0 restriction of data names and data-block codes to 32 characters.

Other options allow the program to write extensive debugging information to a user-specified file, indicating its internal state upon processing each token of input, and to list either a brief summary of how it may be used or its current version number.

5.3.2.1.3. *Limitations of vcif*

Because the program is testing certain properties of character strings within logical lines of a file, it stores a line at a time for further internal processing. If a line contains a null character (an ASCII character with integer value zero), this will be taken as the termination of the string currently being processed, according to the normal conventions in the C programming language for marking the end of a text string. In this case, subsequent error messages may not reflect the real problem. The null character, of course, is not allowed in a CIF.

vcif also interprets syntax rules literally, so a misplaced semicolon might mean that a large section of the file is regarded as a text field and too many or too few error messages are generated. This can make a correct interpretation of the causative errors difficult for a novice user.

5.3.3. Editors with graphical user interfaces

A useful class of editing tool is the graphical editor, where different types of access can be provided through icons, windows or frames, menus and other graphical representations. The availability of standard instructions through drop-down menus makes such tools particularly suitable for users who are not expert on the fine details of the file format. The ability within the program to restrict access to particular regions of the file makes it easier to modify the contents of a CIF without breaking the syntax rules. A small but growing number of such editors are becoming available, such as those described here.

5.3.3.1. *enCIFer*

The program *enCIFer* (Allen *et al.*, 2004) has been developed as a graphical utility designed to indicate clearly to a novice user where errors are present in a CIF, to permit interactive editing and revalidation of the file, and to allow visualization of three-dimensional structures described in the file. In its early releases, it was targeted at the community of small-molecule crystallographers interested in publishing structures or depositing them directly in a structure database. Version 1.0 depended on a compiled version of the CIF core dictionary, but subsequent versions allow external CIF dictionaries to be imported. At the time of publication (2005), development is concentrating on support for DDL1 dictionaries.

Given its target user base, the purpose of the program is to permit the following operations within single- or multi-block CIFs:

(i) Location and reporting of syntax and/or format violations using the current CIF dictionary.

(ii) Correction of these syntax and/or format violations.

(iii) Editing of existing individual data items or looped data items.

(iv) Addition of new individual data items or looped data items.

(v) Addition of some standard additional information *via* two data-entry utilities prompting the user for required input ('wizards'): the *publication wizard*, for entering the basic bibliographic information required by most journals and databases that accept CIFs for publication or deposition; and the *chemical and crystal data wizard*, for entering chemical and physical property information in a CIF for publication in a journal or deposition in a database.

(vi) Visualization of the structure(s) in the CIF.

In all cases where data are edited or added, *enCIFer* can be used to check the format integrity of the amended file.

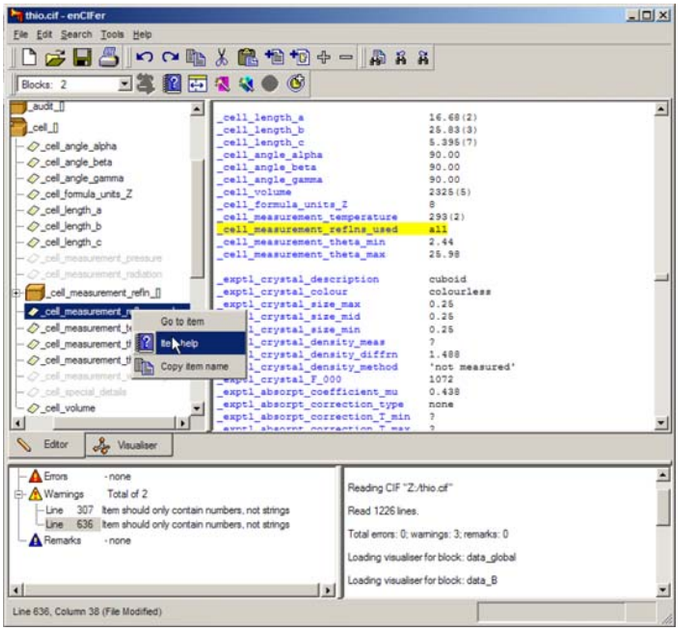


Fig. 5.3.3.1. The *enCIFer* graphical user interface.

5.3.3.1.1. *The main graphical window*

Fig. 5.3.3.1 is an example of the use of *enCIFer* to read and modify a CIF. The figure shows the components of the main window after a file has been opened. Beneath the standard toolbar that provides access to operating-system utilities and to the main functions of the program itself is a task bar (here split over two lines) providing rapid access to a subset of the program's features. Under this are two large panes. The pane on the right is the editing window, where the content of the CIF is displayed and may be modified. The left-hand pane is a user-selectable view by category of the data names stored in the CIF dictionary against which the file is to be validated. At the bottom are two smaller panes. The one on the right logs the session activities and displays informational messages. The left-hand pane lists errors and warning notices generated by the validation system. Errors are labelled by line number, and selection of a specific message (by a mouse double-click) scrolls the content of the main text-editing window to that line number.

Tabs in the middle of the display allow the user to switch rapidly between the editing mode and a visualization of the three-dimensional structures described in the CIF.

These components are described more fully below, followed by a description of the other windows that may be created by a user: the help viewer, the loop editor and the data-entry wizards.

5.3.3.1.2. *The interface toolbar*

This toolbar provides menus labelled 'File', 'Edit', 'Search', 'Tools' and 'Help' that provide the expected functionality of graphical interfaces: the ability to open, close and save files, store a list of recently accessed files, spawn help and other windows, allow searching for strings within the document, allow the user to modify aspects of the behaviour or the look and feel of the program, and provide entry points for specific modes of operation. The most useful of these utilities can also be accessed from icons on the task bar. They are discussed in more detail in the following section.

This main menu is structured in a way familar to users of popular applications designed for the Microsoft Windows operating

system, although the *enCIFer* program runs on a variety of different operating systems and machine hardware platforms. Nevertheless, the use of a common menu style makes the initial use of the program much easier for novice users and allows the program to be effectively used without detailed study of its documentation.

5.3.3.1.3. *The task bar*

The task bar allows rapid one-click access to the standard operations of creating a new document, opening, saving or printing the contents of the current file, copying, cutting and pasting text, searching for specific text within the document, and undoing or redoing previous edits.

Two buttons allow insertion of complete text files. One allows the user to select any file from local or network-mounted file systems. The other imports a specific file (the location of which may be specified by the user through the 'Preferences...' selection of the main 'Edit' menu). While this specific file may contain anything, it is intended to be a template CIF that a user will tailor to meet their own requirements. The default provided with the software is a standard template distributed by the IUCr for use in submitting articles to *Acta Crystallographica*. In either case, the file is imported at the current editing location and is not subject to validation upon input; the user must manually revalidate the file after import.

An icon on the task bar allows the user to run a validation procedure. This icon will be dimmed (indicating that the validation procedure may *not* be run) unless the user has modified the contents of the CIF. Other icons on the task bar behave in the same manner, allowing the procedures with which they are associated to be executed only under appropriate circumstances. Thus, for example, the looped list editor is not invoked unless the user clicks within the reserved word `loop_` in a list header.

Similarly, the 'help' icon in the task bar is dimmed unless the user has selected a data name in the CIF; when this is done, the icon is activated and clicking on it launches a help window containing the CIF dictionary definition of the data item.

The task bar also contains a drop-down menu listing all the data-block names in the current file. When the user selects one of the data-block names, the edit cursor is positioned at the head of the matching data block in the edit window. This is a rapid and efficient way of navigating within large and complex files.

The other buttons provided on the task bar allow the user to: reduce or increase the font size in the editing window; create a new looped list within the loop-editing window; invoke the publication and data-entry wizards; and hide or reveal the dictionary browse window pane.

Users may modify the appearance of the task bar to retain or conceal subsets of these icons, depending on which they find most useful.

5.3.3.1.4. *The main edit pane*

The main edit pane is a text-editing area where the user may directly modify the content of a CIF. Colours and font styles are used to indicate different syntactic elements. The details of the colours and styles may be modified to suit the user.

For the novice user, this is perhaps the most immediately helpful feature offered by this program. When a trailing semicolon is inadvertently lost from an extended text field, typical sequential parsers may interpret succeeding tokens as part of the quoted text and produce misleading error reports. Within the *enCIFer* edit window, all such text is marked up in a specific colour (green by default) so that the fault is much more obvious to the human eye and its source much easier to locate.

Two other typographic cues are used to help the user to trace errors, or to ensure that certain text has been input correctly. Subscripts and superscripts are represented in a smaller typeface (and in a different colour) so that missing delimiter characters are again obvious to the eye. Secondly, some special characters in the conventional CIF encoding (such as Greek letters) are displayed in an appropriate symbol font when the file is first loaded, so that for example the input string `\a` is rendered as α. Note that the backslash character is retained, and that the symbol character is not generated as new text is input or edited. This scheme therefore has some potential for confusion, but is nevertheless helpful in checking that less obvious special codes have been entered correctly.

The user is free to enter arbitrary text in this pane, possibly breaking CIF syntax rules in the process. Only when the revalidation process is manually invoked will the file be rescanned and any errors reported.

5.3.3.1.5. *The dictionary browse pane*

The upper left-hand pane in Fig. 5.3.3.1 illustrates the dictionary browser, an optional graphical view of the contents of the CIF dictionary against which the file is being validated. (The presence or absence of this pane is toggled from an icon in the task bar.) Box icons represent the contents of categories, and the tree of category containers may be expanded or collapsed as desired to show individual items within categories.

A dictionary view is generated for each separate data block in the CIF. Within the dictionary view of an individual data block, those data items present in the data block are shown in bold; other items defined in the dictionary but absent from the current data block appear in a lighter colour.

Within the dictionary browse pane, a user may select (with a click of the appropriate mouse button) a menu of three options which depend on whether the data name is present or absent in the data block. If present, one option positions the cursor in the editing window at the location of the selected data item. If the item is absent from the data block, the user is given the option to paste the data name into the editing window at the current insertion point. The other options (in both cases) are to copy the data name to the clipboard or to open the help window with the CIF dictionary definition of the selected item.

5.3.3.1.6. *The error notification pane and logging area*

The lower left-hand pane of Fig. 5.3.3.1 illustrates typical error notices generated by the parser when the validation process is invoked. At present, the classification of the severity of errors is guided by the editorial requirements of databases and journals, and does not necessarily match the formal errors dictated by the CIF specification. It is likely that this will change in future releases as validation is driven increasingly by the dictionaries rather than by hard-coded subroutines.

A convenient feature is that double-clicking on the line number in the error report relocates the cursor to that line in the editing pane. At present, error messages are listed by line only – they are not grouped by data block.

The user has a small number of options to control error notification. The choice of the maximum number of consecutive error lines to permit before error checking is abandoned is a useful way, especially for novices, to reduce the amount of output generated by severe syntax errors and to focus on repairing individual errors. The user may also specify a file that contains a set of CIF data names which are considered *mandatory* components of a particular file. Absence of any of these items from the current data

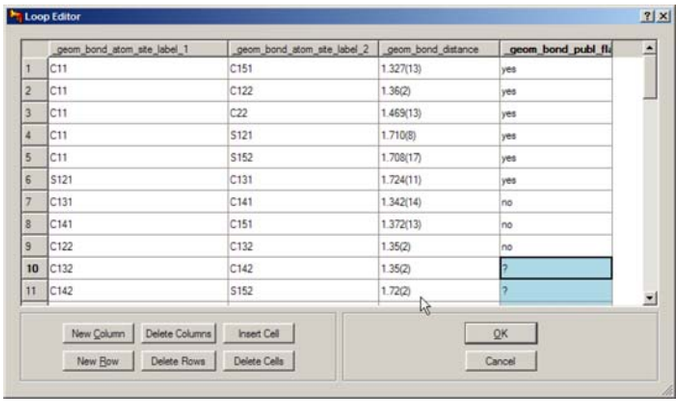


Fig. 5.3.3.2. The *enCIFer* loop editor.

block is flagged as an error. The program log in the lower right-hand part of the program window records the history of the user's interactions with the file during the current editing session.

Information is written to the status bar (the lower margin of the window) to indicate the location by line and column number of the editing cursor.

5.3.3.1.7. *The loop editor*

The program has a useful spreadsheet-style editor for looped lists (Fig. 5.3.3.2). A particular benefit of this style of display is that the spreadsheet cells are arranged in a rectangular grid, so that visual scans can often detect deviations from a pattern of values within a column, thus making it easy to identify placement errors where values have been omitted or inadvertently conjoined. Such errors are not always obvious by direct visual inspection of a CIF, where the layout of a looped list need not follow any regular pattern.

The buttons to add or delete columns allow for the straightforward addition or deletion of data items from the loop. If the user selects the 'New Column' button, a small pop-up window helpfully provides a view of the associated dictionary (in the same hierarchical category-based tree view of the dictionary browser pane) to help the user select the required new data name. The 'Insert Cell' and 'Delete Cells' buttons are convenient tools for the realignment of rows and columns where values have been omitted or misplaced.

The loop editor is invoked from one of two buttons in the task bar, allowing either the creation of a new looped list or the modification of an existing one. As with the application as a whole, there is no dynamic validation of input; the new list must be saved and the entire CIF then manually revalidated.

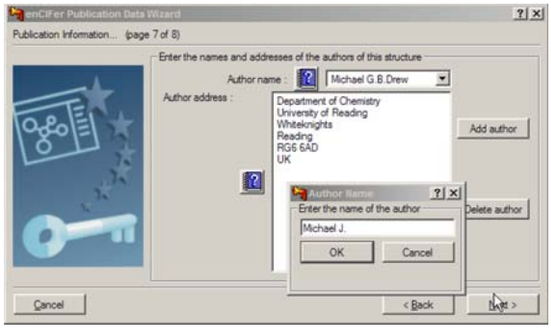


Fig. 5.3.3.3. The *enCIFer* publication data wizard. Information about the title and authors of an article to be submitted for publication is requested through a sequence of linked dialogue boxes.

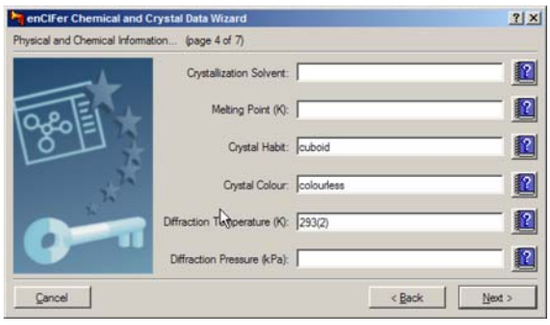


Fig. 5.3.3.4. The *enCIFer* chemical and crystal data wizard.

5.3.3.1.8. *The publication and chemical and crystal data wizards*

The user may invoke data-entry 'wizards', subordinate programs that prompt for particular data items useful for the publication of a crystal structure report or for the deposition of a crystal structure in a database. This is the kind of information that might be requested in the *Notes for authors* for a journal, and it is helpful if the information is routinely requested from inexperienced authors during normal use of the software. The data-entry tools are known as 'wizards' because they will utilize information already in the file.

Hence, as shown in Fig. 5.3.3.3, details of an article's contact author are retrieved from the CIF and used to seed a list of contributing authors. As the address for each author is entered, the program makes each new address available as a stored record for easier input of additional information.

Fig. 5.3.3.4 demonstrates the same approach to encouraging authors to supplement information already in the CIF with related chemical (or crystal) data not usually provided by the CIF generators embedded in crystallographic structure determination programs.

5.3.3.1.9. *The visualization window*

A final useful feature of *enCIFer* is its ability to visualize the three-dimensional structure of molecules described in the data blocks of a CIF. Fig. 5.3.3.5 demonstrates crystal packing with

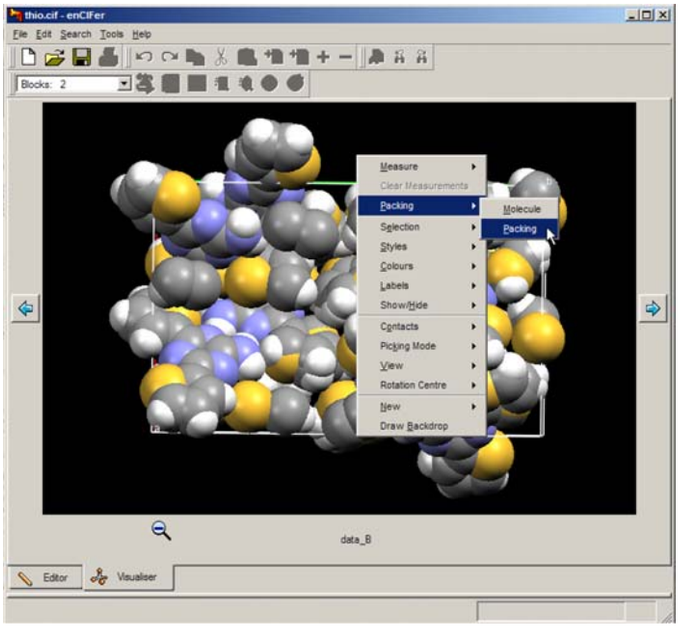


Fig. 5.3.3.5. Visualization of a molecular and crystal three-dimensional structure with *enCIFer*.

a space-filling molecular representation, and the drop-down menu indicates some of the options available to modify the appearance of the graphics. The molecular-graphics library used by the program is part of the larger database interface software package developed at the Cambridge Crystallographic Data Centre. In the present version, the visualizer is run only upon initial parsing of the input CIF, and therefore does not provide an ability to track visually the molecular changes associated with direct modification of the contents of the file.

5.3.3.2. *CIFEDIT*

The *CIFEDIT* program (Toby, 2003) is written in Tcl/Tk (Ousterhout, 1994) and provides an application for viewing and editing CIFs. The code is written in such a way that it can be embedded into larger programs to provide a CIF-editing interface within larger application suites.

The current version of the program is able to validate CIFs against both DDL1 and DDL2 dictionaries, although the DDL2 validation is currently less complete than for DDL1. For example, numeric values are checked against permitted enumeration ranges only for DDL1. Dictionaries are accessed through index files, each of which contains Tcl data structures that point to the location of the definitions in the dictionary file itself and store information such as units and enumeration ranges that can be used for data validation. A utility provided with the program allows a user to generate new index files when new versions of the dictionaries become available. It is intended that dictionary indexing will be incorporated within the main application in the next program release, so that interactive dictionary selection will be possible.

When a CIF is opened, the contents are parsed and validated against one or more user-selected dictionaries. Errors are displayed in a pop-up window and may be written to a file or viewed within the application. The main program window displays the contents of the CIF in two primary panes (Fig. 5.3.3.6). In the left-hand pane, a tree structure shows the data blocks in the file and the data names present in each block. The data blocks may be expanded or collapsed by the user, to present an overview or a detailed view of the data structure of the file. Underneath the icon representing the data block, non-looped data items are listed alphabetically. The figure demonstrates how a single value may be selected in the left-hand pane (`_cell_length_a`) and displayed in the main window. Physical units for the selected quantity are extracted from the corresponding dictionary definition and presented alongside the numeric value. The dictionary definition may also be displayed in a separate pop-up window using the 'Show CIF Definitions' button.

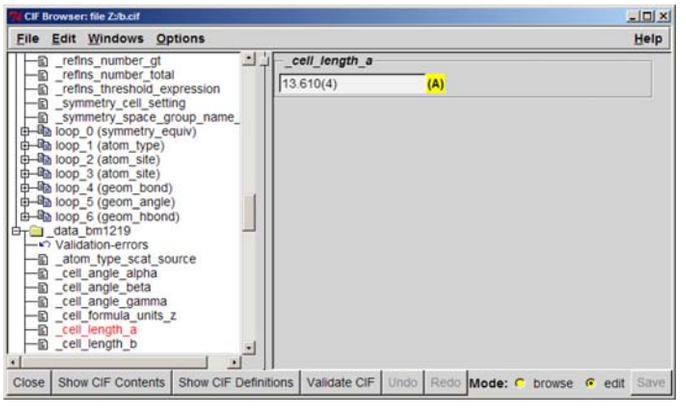


Fig. 5.3.3.6. The use of *CIFEDIT* to display and alter the contents of a CIF; here a non-looped data item is shown.

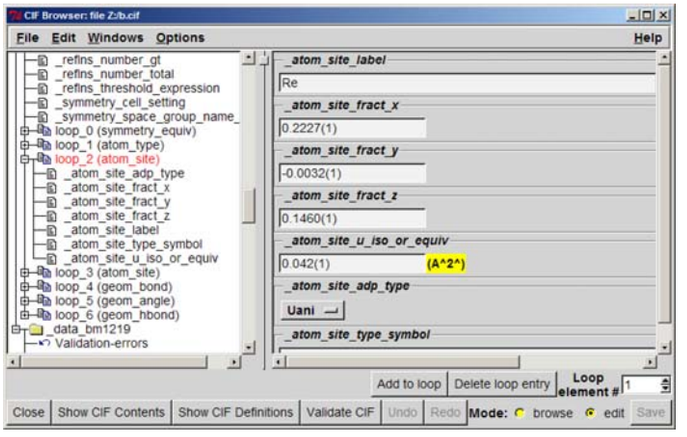


Fig. 5.3.3.7. Row-based loop editing with *CIFEDIT*; here loop_2 (comprising the ATOM_SITE category) has been selected by the user; the editing cursor begins at row 1 of the loop.

The program may be run in two modes: a 'browse' mode, where the selected value is displayed in the main pane, but may not be altered; and an 'edit' mode (as in the example) where the value appears in an editable text widget.

Data loops in the CIF are displayed after the alphabetical list of non-looped items. The loops are numbered sequentially from zero and an indication of the loop category is given in parentheses in the tree-view window. The loop 'branches' of the tree may be expanded or collapsed as the user wishes.

Loops may be viewed and edited in two ways: by row or by column. If the user selects the loop title node in the hierarchical view pane, the loop is presented by row, starting in sequence at row 1 (Fig. 5.3.3.7). Other rows may be selected by using the address box in the lower-right-hand part of the window. Alternatively, if the user selects an individual data name within the loop representation in the hierarchical view, all instances of that data item within the loop are displayed in the main pane. (In practice the number of values shown is constrained to a maximum number that the user may choose, so that the application does not run out of memory if there are very large loops.)

For items with a restricted set of permitted values in the dictionary, the editing function allows the user to select only one of the permitted options *via* a drop-down menu.

While the application is intended to be used in this structured and itemized mode, there is an option to open the entire CIF in a text-editing window if there are errors that cannot be handled in the normal mode. This is not recommended, but is occasionally convenient. While this free-text editing mode is in operation, the ability to modify the file through the structured editing pane is suspended to avoid conflicting changes.

After any change has been made, the user may revalidate the file. This is strongly recommended after making changes in the free-text editing mode.

5.3.3.3. *HICCuP*

The program *HICCuP* (Edgington, 1997) was an early graphical utility developed at the Cambridge Crystallographic Data Centre for interactive editing and validation of a CIF. It is no longer supported, having been replaced by *enCIFer* (Section 5.3.3.1). Nevertheless, it contained some interesting features and is of potential interest to developers using multiple-platform scripting languages. It was implemented in the Python language (van Rossum, 1991) and required that Tcl/Tk (Ousterhout, 1994) be also available on the host computer. The name of the program is an acronym for 'High-Integrity CIF Checking using Python'.

HICCuP was designed to allow users of the Cambridge Structural Database (Allen, 2002) to check structures intended for deposition in the database and therefore included a range of additional content checks specific to this purpose. These could, however, be disabled by the user.

5.3.3.3.1. *Interactive use of the program*

5.3.3.3.1.1. *The control window*

Because *HICCuP* was designed as an interactive tool, upon invocation it presented to the user a *control window* from which CIFs could be selected for analysis and in which summary results of the program's operations were logged. Fig. 5.3.3.8 shows an example of the control window after a single CIF has been loaded.

In the large frame below the file-entry field are listed the data blocks found by the program. The names are highlighted in various colours according to the highest level of severity of errors found within the corresponding data block.

Because the utility was designed for processing large amounts of CIF data for structural databases, it was considered useful to supply a compact visual indicator of the progress of the program through a large file. This takes the form of a grid of rectangular cells, one column for each data block present. Each column contains three cells, which monitor the performance of checks on the file syntax, conformance against a CIF dictionary, and other checks specific to the requirements of the Cambridge Crystallographic Data Centre. As each data block was checked, the corresponding cells were coloured according to the types of error found. Different colours were used to indicate: no errors; structure errors in the initial syntax tests; dictionary errors; or a deviation from certain conventions used by journals and databases in naming datablocks.

The large frame at the bottom of the control window provides a text summary of the same information, listing the number of errors found.

Check boxes and an 'Options…' button allowed some configurability of checks by the user.

5.3.3.3.1.2. *The report frame and edit window*

The user could get more details of the reported errors by clicking on the name of the data block of interest in the control window. The text of the CIF would appear in a new window positioned at the point where the program has detected the first error and a terse statement of the type of error, with a longer explanation of its nature and possible cause, would be given.

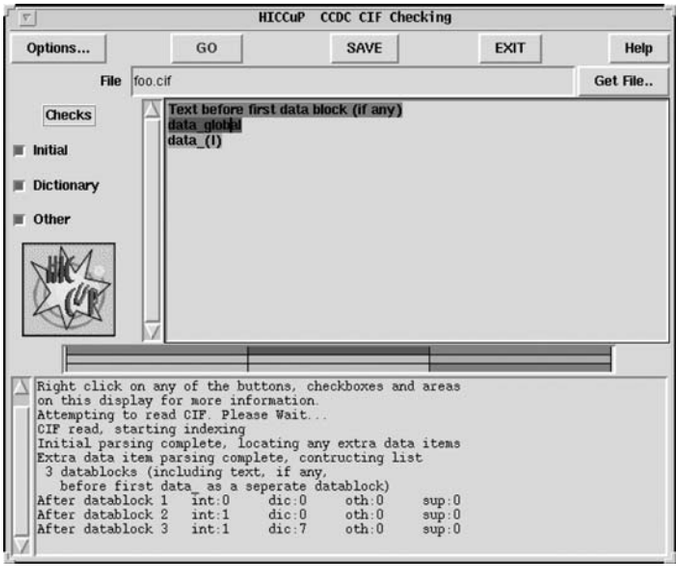


Fig. 5.3.3.8. Control window of the *HICCuP* application.

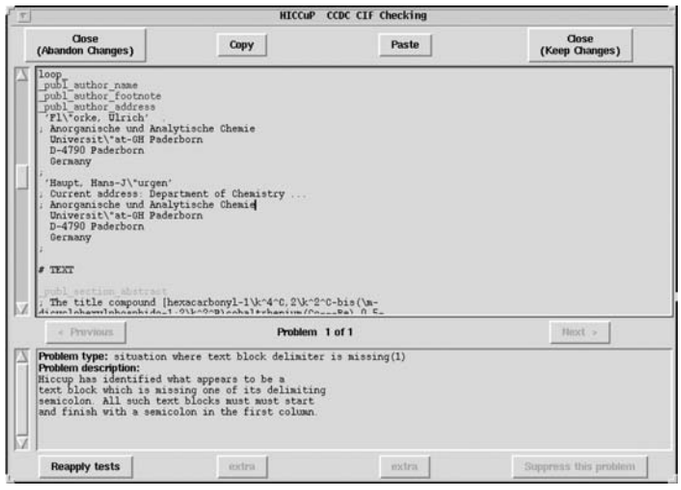


Fig. 5.3.3.9. *HICCuP* edit window and error description.

In the example of Fig. 5.3.3.9, the program has detected that there is a missing text delimiter (a semicolon character), and positions the text in the upper frame at the likely location of the error. The program has attempted to localize the region where the error may have occurred. Because a text field might contain arbitrary contents, including extracts of CIF content, it is impossible to be sure on purely syntactic grounds of the nature of the error. Nonetheless, some heuristic rules serve to identify the author's likely intent in the majority of cases. So, in this example, the user may scan the file contents in the vicinity of the line highlighted by the program and find the error within a few lines (in this example an incorrectly terminated `_publ_author_footnote` entry beginning 'Current address:').

For this example, the more literal *vcif* error analysis provides only the message

```
ERROR: Text field at end of file does not terminate
```

The upper frame in this window is an editable window, so that the user could modify the text and revalidate the current data block. Only when a satisfactorily 'clean' data block was obtained were the changes saved, and the modified data block written back into the original file.

5.3.3.3.1.3. *Dictionary browsing*

An additional useful feature of the program was its interactive link to a CIF dictionary file (Fig. 5.3.3.10). The browser window contains the definition section of the dictionary referring to

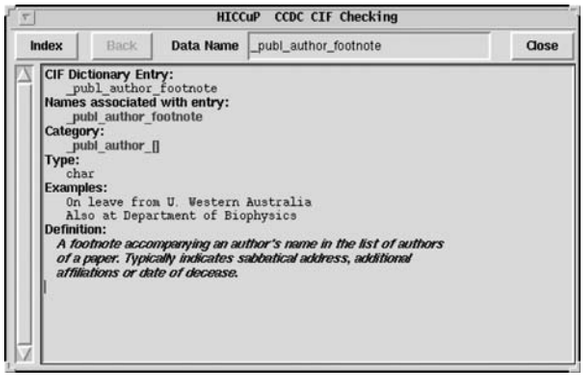


Fig. 5.3.3.10. *HICCuP* dictionary browser window.

the selected data name and hyperlinks to definitions of other data names referred to. Additionally, there is a small text-entry box allowing a specific definition to be retrieved and an 'Index' button to list all available definitions.

5.3.3.3.2. *Options*

As already mentioned, the user could modify the detailed mode of operation of the program. Any or all of the 'initial', 'dictionary' or 'other' checks could disabled.

The 'dictionary' checks could be modified by the user through the 'Options' button of the main control window. The CIF dictionary for validation could be specified; the dictionary itself had to be translated from a source file in DDL format to a Python data structure.

The types of dictionary-based validation supported by the program were:

(i) *List Status* (checking whether a data value should be included in a looped list),

(ii) *Limited Enumeration Options* (checking that a data value is one of the permitted codes where such a constraint exists),

(iii) *Incorrect Enumeration Case* [a special case of (ii), where a data value matches a permitted code except for incorrect alphanumeric case],

(iv) *Enumeration Range* (the data value falls outside the range permitted),

(v) *Value Type (numb or char)* (the data value has the wrong type),

(vi) *List Link Parent* (a data item is present within the data block, but its mandated parent item is not – for example, the data item `_atom_site_aniso_label` should not be present without its parent data item `_atom_site_label`),

(vii) *List Reference* (the required data name used to reference the loop in which the current data name appears is missing),

(viii) *Esd Allowable* (a data value appears to have a standard uncertainty value where one is not expected).

The user could also supply the program with a list of data names that do not appear in the validation dictionary but for which no warning message should be raised. The program normally flagged such nonstandard data names as possible errors and suggested the possible form of a standard data name that might have been intended. This was useful in catching misspellings of additional data items entered by hand.

The program could also be run in a batch mode when the objective was to work through a large volume of CIF data and identify the data blocks that require attention. This mode of operation is particularly useful in databases or publishing houses. In this mode, input is from a named file or from the standard input channel; output is written to standard output or redirected to a results file. The operation of the program may be controlled by the application of various command-line flags.

5.3.3.4. Platform-specific editors

As well as the tools described earlier in this section, which are designed to run under a variety of common operating systems, there are some applications restricted to users of particular types of computer. Here we mention two that run in the popular Microsoft Windows environment on personal computers.

5.3.3.4.1. *beCIF*

The Windows program *beCIF* (Brown *et al.*, 2004) is still in prototype. It is a DDL1-dictionary-driven CIF manipulation tool that does not require detailed knowledge of CIF or dictionary

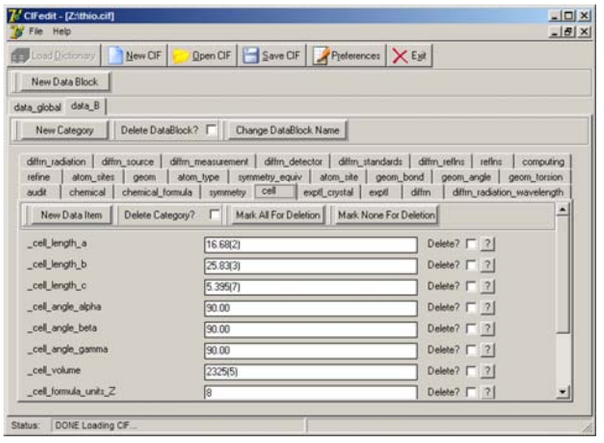


Fig. 5.3.3.11. A category view in the *beCIF* editor of a CIF with navigation by tabs.

structures. It provides a rather different view of the contents of a CIF from the applications discussed above through an interface that will be familiar to users of Microsoft Windows applications. When the application is opened, the user is prompted to provide the location of a CIF dictionary (at any one time, only a single dictionary file may be loaded). This dictionary is loaded into memory and used to validate CIFs upon input. As a data file is read, discrepancies from the types and value ranges permitted by the dictionary are listed in an information window.

The file contents are presented in a number of panels, one per dictionary category, between which the user may navigate by selecting the tab with the desired category name (Fig. 5.3.3.11).

At the highest level, tabs allow the user to choose the data block of interest. Buttons are provided to delete a data block entirely, to rename it or to create a new data block.

Within each data block, the user may add new categories. Again, to help the novice user, when the button 'New Category' is selected, a list of only those categories described in the current dictionary but absent from the current data block is presented to the user. Each category present in the data file is accessed through its own tabbed display panel.

Where the category contains non-looped data items, values may be edited within individual text widgets; data items may be removed by selecting the adjacent check box; or new data items may be added by selecting the 'New Data Item' button to create a dialogue box offering a choice of the remaining data items in the dictionary category. Against each data item a button provides access to a pop-up window containing the relevant dictionary definition.

For a category with looped data, the contents are displayed in a spreadsheet-style representation, with columns headed by the matching data name and rows numbered for convenience (Fig. 5.3.3.12).

The changes requested to the CIF are only effected when the user selects the 'Save CIF' button. Unlike many other of the CIF editors previously discussed, this program does not make any effort to retain the initial ordering of the input data, nor does it preserve comments. The edited CIF may therefore be superficially very different from the input file; however, the only significant differences in content will be those introduced through use of the editing functions within the application.

5.3.3.4.2. *printCIF for Word*

The tools described so far emphasize the data content of a CIF. *printCIF for Word* (Westrip, 2004), on the other hand, was commissioned to help prospective authors of structure reports in the

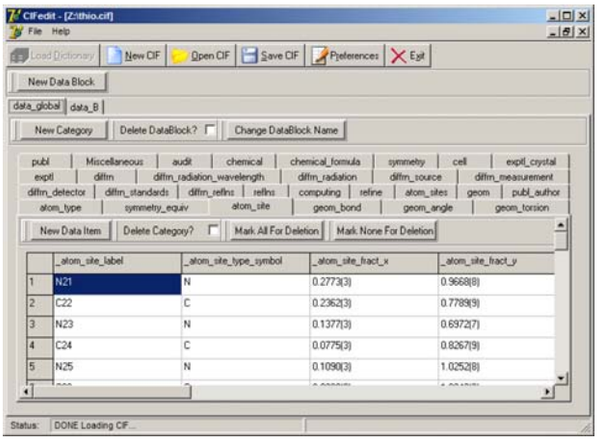


Fig. 5.3.3.12. Representation by the *beCIF* editor of looped data within a category (here ATOM_SITE) in spreadsheet style.

IUCr journals to visualize and prepare for publication complete papers submitted in CIF format. Chapter 5.7 describes the workflow and processing of such submissions. Here is given a brief description of the use of the *printCIF* software from an author's viewpoint.

This application also differs from others discussed in this chapter in that it is rather specific to a particular program environment, being written as Visual Basic macros embedded in a Microsoft *Word* template document. Efforts are under way to provide versions that can run with other word processors. Nevertheless, *Word* is currently sufficiently widespread that the utility is likely to be of use to a large community.

Typically the author begins by double-clicking on the icon associated with the printcif.dot template file. The initial macros are loaded and the author is prompted to provide the location of a CIF. As the CIF is imported into the application, the data items that will be used in the publication are extracted and converted into a rich-text format (RTF) representation. For extended text fields, this RTF content may be edited directly in the word-processing environment; this makes it easy for authors to compose and edit continuous text in a familiar way. Numeric and brief textual data items from the CIF are processed and presented in read-only fields in the manner in which they will appear in the journal, often as entries in a table or as a list of brief experimental details. These fields may not be edited within the RTF representation; if it is necessary to change these, the author must modify the data value in the CIF itself. To assist the author, the contents of the CIF are opened in a text-editor window alongside the formatted representation. The CIF and RTF representations are linked; if the author selects text in the RTF window, the corresponding CIF data item is highlighted within the text-editor window (Fig. 5.3.3.13).

The advantages to the author of editing in RTF format are that existing text may be cut and pasted from other applications, and formatting features, such as subscript or superscript text, Greek letters and other special symbols, may be entered through the word-processor's menu-driven interface, rather than by use of the rather unmemorable ASCII codings used in CIF.

The major disadvantage is the need to recognize that two versions of the file, both editable, are accessible at the same time; and care must therefore be taken to ensure that conflicting changes are not made, and that the author is aware of which version is currently the master. The function 'Update CIF using RTF' (in the toolbar of the CIF editing window) will reimport into the CIF all the editable content from the RTF window, replacing any existing data items.

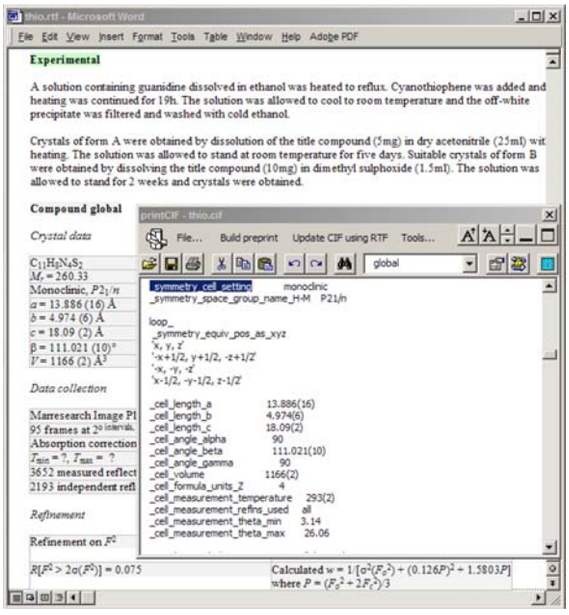

thio.rtf - Microsoft Word
File Edit View Insert Format Tools Table Window Help Adobe PDF
Experimental
A solution containing guanidine dissolved in ethanol was heated to reflux. Cyanothiophene was added and heating was continued for 19h. The solution was allowed to cool to room temperature and the off-white precipitate was filtered and washed with cold ethanol.
Crystals of form A were obtained by dissolution of the title compound (5mg) in dry acetonitrile (25ml) with heating. The solution was allowed to stand at room temperature for five days. Suitable crystals of form B were obtained by dissolving the title compound (10mg) in dimethyl sulphoxide (1.5ml). The solution was allowed to stand for 2 weeks and crystals were obtained.
Compound global
Crystal data
printCIF - thio.cif
File...
Build preprint
Update CIF using RTF
Tools...
global
_symmetry_cell_setting monoclinic
_symmetry_space_group_name_H-M P21/n
loop_
_symmetry_equiv_pos_as_xyz
'x, y, z'
'-x+1/2, y+1/2, -z+1/2'
'-x, -y, -z'
'x-1/2, -y-1/2, z-1/2'
_cell_length_a 13.886(16)
_cell_length_b 4.974(6)
_cell_length_c 18.09(2)
_cell_angle_alpha 90
_cell_angle_beta 111.021(10)
_cell_angle_gamma 90
_cell_volume 1166(2)
_cell_formula_units_Z 4
_cell_measurement_temperature 293(2)
_cell_measurement_reflns_used all
_cell_measurement_theta_min 3.14
_cell_measurement_theta_max 26.06
Data collection
Refinement
Refinement on F²

Fig. 5.3.3.13. The dual RTF/CIF editing windows in the *printCIF for Word* application. In this example, the author has selected the word 'Monoclinic' in the read-only table of crystal data; the corresponding CIF data item `_symmetry_cell_setting` is highlighted in the CIF window, where it may be edited.

The complementary function, 'Build preprint', creates a fresh copy of the preprint representation of the document in RTF format.

A number of options are available to modify the preprint that is generated (for example, by printing a complete list of the geometry included in the CIF rather than just the items flagged for publication; or listing the atomic coordinate data). The general style is that of *Acta Crystallographica Section C* and *Section E*; nevertheless, the application may be useful to users who do not intend to submit to these journals but who wish to produce an attractive representation of the content of their CIFs.

Utilities are provided to create tables in the RTF environment suitable for embedding in the CIF, to browse the contents of the CIF core dictionary and to validate the syntax of the CIF. The application is not dictionary-driven, however, and does not carry out detailed consistency checks. It is therefore best considered as an aid to publication, to be used alongside data-centric editors and validation tools such as *enCIFer*.

A particularly useful self-documenting feature of *printCIF for Word* is that the User Guide is automatically opened when the application is started, before a CIF is loaded.

5.3.4. Data-name validation

In a CIF, a data name (a character token beginning with an underscore character, _) is an essential handle on an item of data within a data block. Equipped only with knowledge of the data names appearing in a CIF, a user may extract, reorder or query the information content of the file. Such manipulations require no prior knowledge of the semantic content of the data. However, for most practical applications it is important to know the meaning attached to data names, and CIF dictionaries provide the mechanism for associating a data name with its intended meaning for an application. It is therefore valuable to be able to check whether data names in a CIF match those defined in a dictionary file. It is also valuable to check the consistency of the data names listed in the dictionary file itself; since this will be used by external applications to validate data names, it is essential that it be internally consistent.

Hence there is a real need for a utility to validate data *names* – effectively a CIF spelling checker.

5.3.4.1. *CYCLOPS*

The program *CYCLOPS* (Hall, 1993; Bernstein & Hall, 1998) was written specifically to address the problem of validating CIF data names. Its use extends beyond simply identifying data names in a CIF data file and checking that they are defined in a dictionary. Any ASCII file may be input, allowing for the checking of CIF data names in any text documents or program source.

The program was originally written in Fortran as an aid to ensuring that the original core CIF dictionary was free from data-name errors; subsequently it was extended to be able to read multiple dictionaries in DDL1 and DDL2 formats, and to resolve data-name aliases across multiple dictionaries. The extended version was written with the library routines of the *CIFtbx* toolkit (Hall & Bernstein, 1996) described in Chapter 5.4 and is distributed as an example application with *CIFtbx*. The description below refers to this extended version, also known as *CYCLOPS2*.

5.3.4.1.1. *Operation*

The program determines the dictionary (or list of dictionaries) against which to validate the input text file (see below for the method of passing such information to the program). It opens each dictionary in turn and stores all data names defined in the dictionaries. Where the same name is defined in multiple dictionaries, the behaviour is determined by a command-line switch.

The text file is then input and parsed for candidate data names. Because the program is designed to check potential data names embedded in ordinary text files, it is not sufficient to apply the CIF parsing rule of a white-space-delimited character string beginning with an underscore character. Instead, character strings are sought that begin with an underscore optionally preceded by white space or one of the characters `,.([{</\|'":*` and followed by white space, one of the characters `,.)]}>/\|'"-=?!;:` or by the end of a line.

For each candidate data name found in this way, matching data names in the stored list are identified in one of three ways:

(i) If the data name is not preceded by the asterisk character `*` and it does not end with the underscore character _, then search for an identical match.

(ii) If the data name ends with the underscore character _, then search for a match in the dictionary where the leading characters in the dictionary name are the same as all the characters in the data name found in the text. For example, the text `_atom_site.label_` would match the mmCIF dictionary entry **`_atom_site.label_alt_id`**.

(iii) If the data name is preceded by the asterisk character `*`, then search for a match in the dictionary where the trailing characters in the dictionary name are the same as all the characters in the data name found in the text. The first match found in the dictionary is accepted. For example, the text `*_alt_id` would match **`_atom_site.label_alt_id`**, or, if that name had not been in the dictionary, **`_struct_conn.ptnr1_label_alt_id`**. If one of the searches succeeds, add the line number of the data name to a list attached to the dictionary name. Up to 19 line numbers are retained for each dictionary name (the first ten matches and the last nine).

If no match is found, the unmatched data name is added to the list of unmatched names, along with the appropriate line number. If a data name has been misspelled it will be caught at this step.

When the text file has been processed, a validation report file is output containing the alphabetically sorted list of unmatched names and line numbers, followed by the sorted list of names from all dictionaries that are used within the text. If requested, this is followed by the sorted list of names from all dictionaries that are not used within the text in the file. If a data name has an alias defined in the dictionaries, a warning about the existence of the alias is given. If more than one dictionary has been used, the source dictionary is identified for each data name. An example of the output from *CYCLOPS* is shown in Fig. 5.3.4.1.

5.3.4.1.2. *Invocation of the program*

CYCLOPS is generally invoked from a command line that specifies the input and output file names and the dictionary files against which to validate the input. However, because the program is portable across a wide range of operating systems, there is substantial flexibility in the way in which it may be invoked. Under a Unix-like operating system, the program may typically be called with a command such as

`cyclops -i` *`infile`* `-o` *`outfile`* `-d` *`dictfile`*

where *infile* is the name of the input file for validation, *outfile* is the file to which the detailed output of the program is written and *dictfile* is a dictionary file.

A more complete set of options available in a Unix-like operating environment is

`cyclops` [`-i` *`infile`*] [`-o` *`outfile`*] [`-d` *`dictfile`*] [`-p` *`priority`*] [`-f` *`cmndfile`*] [`-c` *`catck`*] [`-v` *`verbose`*] [`-s` *`short`*]

where the options are as follows:

-i specifies the name of the input file, *infile*.

-o specifies the name of the output file, *outfile*.

-d specifies the name of the dictionary file, *dictfile*. For compatibility with the original version of the software, the dictionary file may be *either* a CIF dictionary or a list of file names. That is, it may contain dictionary definitions in DDL format or (if the file begins with the characters `#DICT`) it may contain a list of dictionary file names to be entered. As implied by this last statement, multiple dictionaries may be specified to the program.

-p specifies the priority that should be assigned if multiple definitions for the same data name are encountered when multiple dictionaries are accessed. The permitted values are: *first* (the default), in which the first of duplicate definitions to be loaded takes priority; *final*, in which the last takes priority; and *nodup*, in which an instance of a duplicate definition should be treated as a fatal error.

-f specifies the name of a command file *cmndfile* that contains additional directives to the program.

-c is a flag indicating whether an error message should be raised if a data name has been assigned a category different from the leading portion of the data name itself. The Boolean variable *catck* may take the values 't', '1' or 'y' for *true*, 'f', '0' or 'n' for *false*.

-v is a flag indicating whether a verbose listing of unreferenced data names should be generated. The Boolean variable *verbose* may take the same values for *true* or *false* as above.

-s is a flag indicating whether the output should be short (*i.e.* restricted to items not in dictionaries). The Boolean variable *short* takes the same values as above.

For the flags expecting Boolean values, the default is 'f' (*false*).

If no input or output file names are specified, the program will read from the standard input channel or write to standard output,

```
           CYCLOPS Check List
           ------------------
   Dictionary data names  = 2244
   New data names in text =    4
   [1]  Dictionary cif_core.dic 2.0.1 data names =  624
   [2]  Dictionary cif_mm.dic 0.9.0 data names = 1620

Data names NOT in Dictionary           Line Numbers

_blat1 . . . . . .                     9   11   94  96
                                       181  183  290  296
_blat2 . . . . . .                     13   15   98  100
                                       185  187  287  293
_dummy_test  . . . . . .                5    7   90  92
                                       177  179  201
_rubbish_here. . . . . .               431

[1]  Dictionary cif_core_2.0.1.dic
[2]  Dictionary cif_mm.dic
                                       Line Numbers

[2] _atom_site.calc_attached_atom      413
[1] = _atom_site_calc_attached_atom    412
[2] _atom_site.calc_flag  . . . . .    410
[1] = _atom_site_calc_flag             409
[2] _atom_site.fract_x  . . . . . .     38   44   50  390
[1] = _atom_site_fract_x               389
[2] _atom_site.fract_y  . . . . . .     39   45   51  394
[1] = _atom_site_fract_y               393
[2] _atom_site.fract_z  . . . . . .     40   46   52  398
[1] = _atom_site_fract_z               397
[2] _atom_site.id   . . . . . .         37   43   49  386
[1] = _atom_site_label                 385
[2] _atom_site.thermal_displace_type   406
[1] = _atom_site_thermal_displace_type 405
[2] _atom_site.type_symbol  . . . .    416  420  424  428
                                       434  438  442  450
[1] = _atom_site_type_symbol           415  419  423  427
                                       433  437  441  449
```

[later in the validation output file, showing the transition to unreferenced data names . . .]

```
[1] _symmetry_cell_setting  . . . .    319
[2] = _symmetry.cell_setting           320
[1] _symmetry_space_group_name_H-M     323
[2] = _symmetry.space_group_name_H-M   324
[1] _symmetry_space_group_name_Hall    327   445
[2] = _symmetry.space_group_name_Hall  328   446

[1]  Dictionary cif_core_2.0.1.dic
[2]  Dictionary cif_mm.dic
                                  Names Not Referenced

[2] _atom_site.aniso_B[1][1]
[2] _atom_site.aniso_B[1][1]_esd
[2] _atom_site.aniso_B[1][2]
```

[. . . portion of output omitted . . .]

```
[2] _atom_site.aniso_U[3][3]_esd
[2] _atom_site.attached_hydrogens
[1] = _atom_site_attached_hydrogens
[2] _atom_site.auth_asym_id
[2] _atom_site.auth_atom_id
[2] _atom_site.auth_comp_id
[2] _atom_site.auth_seq_id
[2] _atom_site.B_equiv_geom_mean
[1] = _atom_site_B_equiv_geom_mean
[2] _atom_site.B_equiv_geom_mean_esd
[2] _atom_site.B_iso_or_equiv
[1] = _atom_site_B_iso_or_equiv
[2] _atom_site.B_iso_or_equiv_esd
```

[. . . remainder of output omitted . . .]

Fig. 5.3.4.1. Sample output from *CYCLOPS*. The output has been edited and reformatted slightly to fit into the present column width.

respectively. The special character hyphen ('-') may also be supplied as an argument to '*-i*' or '*-o*' to indicate standard input or standard output.

Finally, if the operating system supports the passing of environment variables to a program, the names of the input file, output file and dictionary file may be passed through the values of `$CYCLOPS_INPUT_TEXT`, `$CYCLOPS_VALIDATION_OUT` or `$CYCLOPS_CHECK_DICTIONARY`, respectively.

5.3.5. File transformation software

This section describes a number of applications that transform an input CIF either to another CIF that contains a subset of the original contents or to other formats suitable for use with general processing tools. (Conversion to other crystallographic data formats is not discussed here.)

5.3.5.1. *QUASAR*: a data extractor

The oldest CIF manipulation program is *QUASAR* (Hall & Sievers, 1993), which was described as the prototype CIF application in the original standard specification paper (Hall *et al.*, 1991). Much of the functionality of *QUASAR* has now been included in the *cif2cif* program (Section 5.3.5.2). However, it remains useful as an application in its own right, and so is briefly described here.

5.3.5.1.1. *Purpose*

The program was designed to read a *request list* of data names, to locate the associated data in an input CIF and to output the data in the order of the request list. The output retains local conformance to CIF syntax rules, but the output file may not be strictly CIF conformant. For example, the same data can be requested multiple times and will be reproduced as often as requested in the output stream, a feature forbidden within a legal CIF.

5.3.5.1.2. *Mode of operation*

Written as a pure Fortran77 application, *QUASAR* requires three data streams: a file containing the request list, an input CIF and an output file. In an operating system such as Unix, it is convenient to attach the request list to the standard input channel; the first two lines of the input stream then take the form `star_arc_`*`infile`* and `star_out_`*`outfile`*, where *infile* and *outfile* are the file names of the input and output files, respectively.

The assignment of an output file may be replaced by a line containing `star_log`. When this is done, the program will test the syntactic validity of the input CIF and write any error messages to the standard output channel. In this mode the program may be used as a syntactic validator, although it is more tolerant of certain syntactic errors than *vcif* (Section 5.3.2.1).

5.3.5.1.3. *The request list*

Fig. 5.3.5.1 is an example request list, intended to highlight some of the special features of the way the program operates. Fig. 5.3.5.2 shows an example CIF against which this request list will be tested; Fig. 5.3.5.3 shows the output. Both figures have been modified slightly to fit on the printed page; they are derived from the sample files distributed with the program.

The request list begins with directives specifying the input and output file names (qtest.cif and qtest.out, respectively). The file may contain comments prefaced by a hash character `#`; this is a useful feature for annotating a request list. Another use for such comments is seen in the standard request list distributed to authors for papers published in *Acta Crystallographica*. Here, data names that are *not* normally published are hidden within the request list as comments and may be activated if they occur in a `publ_manuscript_incl_extra_item` loop within a CIF (see Section 5.7.2.3).

```
star_arc_qtest.cif
star_out_qtest.out

data_    #<< wild-card block name - accepts first

# request all fractional coord items
   _atom_site_fract_
   _atom_site_label
# capitals to test case insensitivity
   _atom_site_aniso_LABEL
# request something that is not in the CIF
   _dummy
   _atom_site_aniso_U_11

data_P6122
_        #<< this requests all data in this block
```

Fig. 5.3.5.1. An example request list for *QUASAR*.

```
data_P6122

loop_
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
   # capitals to test case insensitivity
_atom_type_scat_dispersion_REAL
_atom_type_scat_dispersion_imag
_atom_type_scat_source
  S   0  6   .319  .557
        Int_Tab_Vol_III_p202_Tab._3.3.1a
  O   0  6   .047  .032
        Cromer,D.T._&_Mann,J.B._1968_AC_A24,321.
  C   0  20  .017  .009
        Cromer,D.T._&_Mann,J.B._1968_AC_A24,321.
  RU  0  1  -.105   3.296
        Cromer,D.T._&_Mann,J.B._1968_AC_A24,321.

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
_atom_site_type_symbol
  s   .20200  .79800  .91667  .030(3)  Uij  ?  ?  s
  o   .49800  .49800  .66667  .02520  Uiso  ?  ?  o
  c1  .48800  .09600  .03800  .03170  Uiso  ?  ?  c

loop_
_atom_site_aniso_label
_atom_site_aniso_U_11
_atom_site_aniso_U_22
_atom_site_aniso_U_33
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_23
_atom_site_aniso_type_symbol
  s .035(4) .025(3) .025(3) .013(1) .000 .000 s
```

Fig. 5.3.5.2. Example CIF for demonstrating the use of *QUASAR*.

```
data_P6122

loop_
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_label
  .20200 .79800 .91667 s
  .49800 .49800 .66667 o
  .48800 .09600 .03800 c1

loop_
_atom_site_aniso_label
_dummy           # requested item not present
_atom_site_aniso_U_11
  s ? .035(4)

#                -----end-of-data-block-----

data_P6122

loop_
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_scat_dispersion_REAL
_atom_type_scat_dispersion_imag
_atom_type_scat_source
  S  0  6  .319  .557
        Int_Tab_Vol_III_p202_Tab._3.3.1a
  O  0  6  .047  .032
        Cromer,D.T._&_Mann,J.B._1968_AC_A24,321.
  C  0 20  .017  .009
        Cromer,D.T._&_Mann,J.B._1968_AC_A24,321.
  RU 0  1 -.105 3.296
        Cromer,D.T._&_Mann,J.B._1968_AC_A24,321.

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
_atom_site_type_symbol
  s  .20200 .79800 .91667 .030(3) Uij  ? ? s
  o  .49800 .49800 .66667  .02520 Uiso ? ? o
  c1 .48800 .09600 .03800  .03170 Uiso ? ? c

loop_
_atom_site_aniso_label
_atom_site_aniso_U_11
_atom_site_aniso_U_22
_atom_site_aniso_U_33
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_23
_atom_site_aniso_type_symbol
  s .035(4) .025(3) .025(3) .013(1) .000 .000 s

#                -----end-of-data-block-----
```

Fig. 5.3.5.3. Result of running *QUASAR* with the example request list of Fig. 5.3.5.1 on the CIF listed in Fig. 5.3.5.2.

The request list must specify the data block from which the requested data are to be extracted. Multiple data blocks may be requested in the same file. An entry '`data_`' operates as a wild card and indicates that requests should be served from the next data block encountered. In the example above, the first group of requests will be met from the first data block in the CIF; the second set from the data block named 'P6122' (if present).

5.3.5.1.4. *Output from QUASAR*

The body of the request list is a series of data names. Where a data name appears in the CIF, it will be extracted with its associated data value or values. The user need not have prior knowledge of whether a data item occurs in a looped list or not: *QUASAR* will automatically retrieve the matching values and construct a loop header if necessary. However, because the requests are served in the exact order in which they occur in the file, data items in the same list in the input CIF may be extracted into different lists upon output. Although this breaks the semantic association between items grouped in the same list (especially for CIFs described by the DDL2 relational scheme), it is a syntactically valid construction and may be a valuable feature for some processes.

5.3.5.1.4.1. *Treatment of missing data*

When a requested data item is absent from the CIF, *QUASAR* will nevertheless emit a data name with a corresponding value of '?', the conventional CIF value of null type for 'unknown quantity'. A CIF comment is also generated by *QUASAR* to indicate that the entry was missing from the input CIF. If the missing data name is found between data names that have multiple values and that occur in the same looped list, it is assumed that the missing data name should be associated with the same looped list, and it will be emitted in the loop header; the integrity of the list is then satisfied by emitting a column of unknown values. Note how this behaviour differs from that of the generic STAR File extraction utility *Star_Base* (Spadaccini & Hall, 1994), which silently ignores missing data items. However, it is a useful behaviour for applications that depend on finding a specific data item in their processing stream, even where its value is unknown.

5.3.5.1.4.2. *Matching data names*

As with the specification of data-block names, the data names in the request list may have a trailing underscore. Where this is the case, *QUASAR* will retrieve all data items where the data name starts with the specified string. For example, a request for '_atom_site_' will extract *all* data names starting with '_atom_site_'. The special case of an isolated underscore character '_' matches *all* data names present in the current data block.

5.3.5.1.4.3. *Case sensitivity*

The example demonstrates the way in which the application handles the case insensitivity of a requested data item. Data names are converted internally to a lower-case representation, both from the request list and the input CIF. Matches are therefore determined in a case-insensitive manner. However, if a data name is present in the CIF, its original case is retained on output. This permits the computationally irrelevant but cosmetically useful retention of capitalization as used in canonical CIF dictionary definitions. Where the requested data name is absent, the output is all lower-case.

5.3.5.2. *cif2cif*

cif2cif (Bernstein, 1998) is a program built with the *CIFtbx* toolkit (Chapter 5.4) to copy a CIF while checking data names against dictionaries, optionally reformatting numbers to maintain standard uncertainties within a specified range. The output CIF may contain a subset of the data in the original CIF according to a request list, in the manner of *QUASAR* (Hall & Sievers, 1993).

The program was built as a sample application using *CIFtbx* routines and grew out of requirements from several sources.

5.3.5.2.1. *Operation*

5.3.5.2.1.1. *Copying*

In its simplest application, the program copies a CIF from the standard input channel to standard output. The copy is not verbatim (standard utilities of the computer operating system should be used for that purpose), but the output CIF differs from the input only in the following respects: some comments are deleted; lines in the input longer than 80 characters are wrapped to 80 characters or less; white space between tokens may be altered, especially in an attempt to align entries in looped lists in a cosmetically pleasing manner. While none of these changes should affect robust CIF-parsing applications, they are nevertheless useful in imposing a uniform style of presentation for browsing in a text editor or other human-readable framework.

5.3.5.2.1.2. *Constraining standard uncertainties to specified ranges*

Some journals require that standard uncertainties in experimental values should be quoted within a specified range. Typically the standard uncertainty (s.u.) should be quoted as an integer in parentheses, modifying the last place or two of decimals in the experimental data, and with a value between 2 and 19. *cif2cif* permits s.u. values in the ranges 1–9, 2–19 or 3–29, selectable by a command-line switch. The effect of applying the 'rule of 19' would be to change a value of 1.458(1) in the input CIF to 1.4580(10) in the output.

5.3.5.2.1.3. *Dictionary validation*

cif2cif will open one or more CIF dictionary files as it copies the input CIF and identify certain classes of error against the dictionary definitions. The conditions that will raise an error are an unrecognized data name or a wrong data type. The program will also optionally indicate a warning if a data name has been assigned a category different from the leading portion of the data name – this may indicate an inconsistency within the dictionary itself.

5.3.5.2.1.4. *Serving a request list*

cif2cif will extract a subset of the data items contained in a CIF as specified by a request list, in the manner of *QUASAR*. The handling of data names specified in the request list is as described in Section 5.3.5.1.3 above, with the following additional feature. The special string `data_which_contains:` will extract the specified data items from the first data block in which at least one occurs; the block code need not be known in advance.

Some care must be exercised in attempting to extract data from data blocks by context without prior knowledge of the file contents. Consider the following simple example file:

```
data_A
   loop_    _A1
            _A2
             a1 a2 aa1 aa2


data_B
   loop_    _A1
            _B1
             a b aa bb
```

The loop containing **_A1** and **_B1** *cannot* be extracted with a request list of the form

```
data_which_contains:
_A1
_B1
```

because **_A1** occurs in the first data block encountered; the output from *cif2cif* in this example will be

```
data_A
    loop_     _A1
               a1   aa1
#             ---end-of-data-block---
```

The behaviour of the program differs from *QUASAR* in two other small ways. When the request list forces the output data stream to contain the same data-block header more than once, an error message is posted to the standard error channel and the data-block headers in the output stream are annotated with a comment of the form '`#<---- duplicate data block`'. In this case the output file does *not* conform to the CIF syntax rules.

When a data name is requested but no matching data item appears in the output file, *cif2cif* writes an error message to the standard error channel. However, unlike *QUASAR*, which inserts the requested data name in the output stream with an associated value of '`?`' (for *unknown*), *cif2cif* produces *no* output for the requested data item.

5.3.5.2.1.5. *Other features*

Some additional features are of use in special circumstances.

The user may preserve the layout of the contents of looped lists exactly as in the input file, or may ask the program to adjust the layout to a more visually pleasing tabular form.

The user may enable recognition of data-name aliases in the dictionaries used for validation. When the relevant command-line argument is set to *true*, user-supplied data names will be transformed to the canonical forms in the validating dictionary. This would permit, for example, a small-molecule CIF using the core dictionary definitions to be converted to mmCIF format.

The user may prefix each line of output with an identical character string. A typical reason for so doing would be to include a fragment of CIF listing within the body of an email message or some other document. Such an output would not conform to the syntax rules for CIF.

5.3.5.2.2. *Invocation of the program*

cif2cif is another application of the *CIFtbx* library by the same author, and so has a similar user interface to that of *CYCLOPS* (Section 5.3.4.1.2). Under a Unix-like operating system, the program is typically called with a command such as

`cif2cif -i` *infile* `-o` *outfile* [`-q` *reqfile*]

where *infile* is the name of the input file, *outfile* is the output file and *reqfile* is an optional file containing a request list for a subset of the original contents.

A more complete set of options available in a Unix-like operating environment is

`cif2cif` [`-i` *infile*] [`-o` *outfile*] [`-d` *dictfile*] [`-q` *reqfile*] [`-f` *cmndfile*] [`-c` *catck*] [`-a` *alias*] [`-t` *tab*] [`-e` *sulim*] [`-p` *prefix*]

where the options are as follows:

-i specifies the name of the input file, *infile*.

-o specifies the name of the output file, *outfile*.

-d specifies the name of a dictionary file, *dictfile*, against which the existence, type and category of data names are checked. The dictionary file may be *either* a CIF dictionary or a list of file names. That is, it may contain dictionary definitions in DDL format or (if the file begins with the characters **#DICT**) it may contain a list of dictionary file names to be entered. Thus, multiple dictionaries may be specified to the program.

-q specifies the name of the request file, *reqfile*, containing a list of data names (with associated data-block directives) that should be extracted as a subset of the contents of the original file.

-f specifies the name of a command file *cmndfile* that contains additional directives to the program.

-c is a flag indicating whether an error message should be raised if a data name has been assigned a category different from the leading portion of the data name itself. The Boolean variable *catck* may take the values 't', '1' or 'y' for *true*, 'f', '0' or 'n' for *false*.

-a is a flag indicating whether data-name aliases in the validating dictionary should be used to replace user-supplied names by their canonical forms. The Boolean variable *alias* may take the same values for *true* or *false* as above.

-t is a flag indicating whether the output should be reformatted with tabs to produce a regular table layout within looped lists. The Boolean variable *tab* takes the same values as above. If *true*, text is reformatted; if *false*, the original formatting is retained.

For the flags expecting Boolean values, the default is 'f' (*false*).

-e specifies the precision to retain in rounding standard uncertainty values. The permitted integer values are 9, 19 (the default) and 29.

-p takes a string value which is prefixed to every line of output. Every occurrence of the underscore character '_' in the prefix is changed to a space on output.

If no input or output file names are specified, the program will read from the standard input channel or write to standard output, respectively. The special character hyphen ('`-`') may also be supplied as an argument in place of a file name to indicate standard input or standard output as appropriate.

Finally, if the operating system supports the passing of environment variables to a program, the name of the input file may be passed as the value of `$cif2cif_INPUT_CIF`, and likewise the output file, `$cif2cif_OUTPUT_CIF`, dictionary file, `$cif2cif_CHECK_DICTIONARY`, and request file, `$cif2cif_REQUEST_LIST`, may be specified.

5.3.5.3. *ciftex*: translating to a typesetting language

The program *ciftex* (McMahon, 1993) was developed to create files for typesetting the journal *Acta Crystallographica* using the text-formatting language TEX (Knuth, 1986). Details of its use in the journal production process are given in Chapter 5.7. It is discussed here as an example of translating a CIF to some output format where data values are annotated with different text depending on their accompanying data names.

5.3.5.3.1. *Basic operation of ciftex*

The program is designed to act as a filter, typically in a Unix-style environment, reading a CIF on the standard input channel and outputting a modified data stream to standard output. The output is a file of TEX code that is processed by the TEX program to produce a device-independent file describing the content of a formatted typeset document. Further post-processing allows the formatted document to be viewed on the screen or printed.

Each input token (number, character or text string; data name; **loop_** or **data_** keywords) is transformed as it is identified; there is no lookahead and minimal retention of context. The data stream is treated purely syntactically; no transformations are applied on the basis of the supposed meaning of any of the file contents.

```
_cell_formula_units_Z        2
_cell_length_a               8.79(2)
_refine_ls_extinction_coef    .347e4(5)
_chemical_name_common        'copper sulphate'
```

Fig. 5.3.5.4. Sample CIF data input to *ciftex*.

```
\cellz{2}
\nobreak\cella{8.79 (2)}
\extcoeffLarson{0.347 (5) $\times$ $10^{4}$}
\chemcom{copper sulfate}
```

Fig. 5.3.5.5. Output from *ciftex* run on the data of Fig. 5.3.5.4.

5.3.5.3.1.1. *Non-looped data*

For portions of the CIF that are not contained in looped lists, the transformations are trivial. A (*data name*, *data value*) pair is transformed to a TEX macro and its argument. The macro name is determined from an external 'map' file which the program reads at run time; this file associates CIF data names and the corresponding TEX macros through a simple lookup table.

A CIF data value is in most cases passed as the argument to the corresponding TEX macro with few modifications. If the data value is a character string beginning with an integer, full point, hyphen or plus character, it is assumed to be of type 'numb'. A space is introduced ahead of an embedded open parenthesis (to separate a standard uncertainty from its parent value). A leading zero is printed before any bare decimal point. An embedded `E` is taken to indicate exponential notation and the format of the number is accordingly modified.

If the input data value is of type 'char' (*i.e.* is a single token beginning with characters other than those recognized as the leading characters for numerical data; or contains multiple tokens delimited by quote marks or semicolons), the program will search the map file for key values exactly matching each token, and if found will substitute the token by its replacement word or text. If no replacement is specified in the map file, the token is passed unchanged to the standard output channel. This facility was found to be useful in making global substitutions of individual words during file processing, but must be used with care since the substitutions are unconditional, without any reference to context.

Some small examples of typical non-looped data items are shown in Fig. 5.3.5.4 and the corresponding *ciftex* translation based on a map file used for typesetting *Acta Crystallographica Section C* is shown in Fig. 5.3.5.5.

Note the transformations of the numerical arguments and the translation of 'sulphate' to 'sulfate'.

5.3.5.3.1.2. *Looped data*

If the input token is a `loop_` keyword, the program enters a different mode of operation. Looped data may be represented in print either as repetitive lists or in tabular format. There is no indication in a CIF dictionary of the appropriate representation (nor should there be, for what is essentially a matter of presentation) and the choice is made based on a flag associated with each data name in the map file. For non-tabular lists, the structure

```
loop_
    _dataname_1
    _dataname_2
       value_1    value_2
       value_3    value_4
```

```
\settabs 5 \columns
\+\relax & $x$  &   $y$    &  $z$   &$U_{\rm eq}$&\cr
\+Re &0.222 (1) &0.003 (1) &0.146 (1) &0.042 (1) &\cr
\+Co &0.234 (1) &0.139 (1) &0.299 (1) &0.046 (1) &\cr
\+P1 &0.358 (1) &0.222 (1) &0.197 (1) &0.044 (1) &\cr
\+P2 &0.106 (2) &0.051 (1) &0.289 (1) &0.046 (1) &\cr
\+C1 &0.308 (6) &0.029 (6) &0.034 (4) &0.057 (4) &\cr
\+O1 &0.356 (5) &0.044 (5) &0.030 (3) &0.079 (3) &\cr
\+C2 &0.066 (6) &0.039 (6) &0.111 (4) &0.056 (4) &\cr
```

Fig. 5.3.5.6. TEX markup for typesetting a table of atomic coordinates.

is translated to a sequence of TEX codes of the form

```
\macro_one(value_1)
\macro_two(value_2)
\macro_one(value_3)
\macro_two(value_4)
```

In the case of tabulated data, the `loop_` header is translated into a set of table headings and typographic codes are introduced to lay out in columnar format the values in the body of the list. The number of different data names in the loop header is counted and the data values are identified by their position in the loop modulo the total number of data names in the header (in effect, by their 'phase' in the loop). In the simplest case, a TEX command is emitted that builds a table with *n* columns, where *n* is the number of different data names. Then the data values are counted as they are processed. After every *n*th data value, a TEX code is emitted indicating 'end of table row' and a further code is emitted before the next value (if there is one) that means 'beginning of new table row'. In all other cases, a code is emitted signifying 'move to next column'.

Fig. 5.3.5.6 is a simplified extract from a table of atomic coordinates derived from the `_atom_site_` loop in a CIF.

5.3.5.3.1.3. *The ancillary map file*

The translation between a CIF data name and its replacement text in the TEX output file is defined in the external map file. The format of the translation is very simple, as illustrated in Fig. 5.3.5.7.

Each line starts with a CIF data name, which is terminated by a space character. The next character is either '`T`' or '`N`' to indicate whether the output should be tabulated or not. The next character is an arbitrary character from the ASCII character set, and is chosen to collect together data that will appear in the same logical section of the output file. This locator character may be associated, in another ancillary file described below, with additional text for output. The remainder of the line is the replacement text.

In the example supplied, the cell-length parameters map to the TEX macros `\cella`, `\cellb` and `\cellc` (each preceded by a standard TEX macro forbidding a page break immediately before the contents are printed). The details of the publication authors are described by a set of TEX macros that will occur in two different locations in the output file (the authors' names and addresses may be looped together in the location labelled by the character `a`; any explanatory footnotes and email addresses will be printed elsewhere in the paper, at the location labelled `x`). The anisotropic displacement parameters U^{ij} will be printed in a table and the replacement text consists of the TEX codes that will be printed at the head of each column in the table.

The initial text on the line need not be a CIF data name; it may be any other single word. In this case, every occurrence of that word in the input CIF will be replaced by the replacement text.

```
_cell_length_a Ng\nobreak\cella
_cell_length_b Ng\nobreak\cellb
_cell_length_c Ng\nobreak\cellc

_publ_author_name Na\author
_publ_author_address Na\address
_publ_author_footnote NX\aufootnote
_publ_contact_author_email NX\email

_atom_site_aniso_label TU\relax
_atom_site_aniso_U_11 TU{\hfill $U^{11}$ \hfill}
_atom_site_aniso_U_12 TU{\hfill $U^{12}$ \hfill}
_atom_site_aniso_U_13 TU{\hfill $U^{13}$ \hfill}
_atom_site_aniso_U_22 TU{\hfill $U^{22}$ \hfill}
_atom_site_aniso_U_23 TU{\hfill $U^{23}$ \hfill}
_atom_site_aniso_U_33 TU{\hfill $U^{33}$ \hfill}
```

Fig. 5.3.5.7. Example map file for use with *ciftex*.

```
#[:\newif\ifproof \prooftrue
#]:\iftwocol\vfill\enddoublecolumns\fi
#a:\pretolerance1000\parskip0pt\tolerance5000
#a:\vskip10pt
#g:
#g:%%%
```

Second, the replacement text should be externalized as much as possible. The use of map and format files means that the same basic program can be used for formatting according to any set of typographic rules; only the ancillary files need to be modified. In the current version of *ciftex*, the program performs some replacements internally; an objective of further development is to remove this function from the program and to externalize it either in more sophisticated table lookup files or in separate methods modules.

Third, the concept of replacement should be abstracted as much as possible. The software was written initially with the objective of replacing data names with TEX macros. Experience suggests that a generic transformation program could be written with the philosophy of replacing data names and data values by directives implemented before and after the occurrence of the data value, and as events upon its first, last and intervening occurrences. Such 'directives' and 'events' could be mapped to arbitrary replacement strings in any markup scheme, such as SGML, XML, HTML, TEX, LATEX or commercial word-processing encodings.

5.3.6. Libraries for scripting languages

Recent years have seen a great increase in the popularity of scripting languages – broadly, computer languages where the source code is run-time interpreted and that have powerful facilities for interacting with other processes and operating-system utilities. In part this is because such languages lend themselves to rapid prototyping; but the powerful features of languages such as Perl (Wall *et al.*, 2000), Python (van Rossum, 1991) and Tcl/Tk (Ousterhout, 1994) make it entirely feasible to create and maintain complex programs that can run efficiently. Several of the applications discussed in this chapter make effective use of one or more of these languages, either solely or alongside more traditional compiled languages.

As the scripting languages have become more powerful, they have also acquired more structure, so that authors now frequently build libraries (or 'modules') of functions or subroutines that can be re-used in a range of applications. This is a welcome development, for the availability of public libraries not only reduces the effort required to develop new applications, but also goes a long way towards establishing common application programming interfaces that help to standardize the way in which software from different sources is developed.

In this section two available libraries of this type are reviewed: *STAR::Parser* and *PyCifRW*.

5.3.6.1. *STAR::Parser* and related Perl modules

A collection of Perl modules has been developed at the San Diego Supercomputer Center (Bluhm, 2000) to provide basic library routines for object-oriented manipulation of STAR files with restricted syntax appropriate to CIF applications. The module name suggests that a more complete STAR implementation may be considered in future developments, but at present the modules do not handle nested loops or the inheritance of data values from global blocks. Indeed, they are still rather limited in scope; nevertheless, for the programmer wishing to prototype CIF applications in Perl, they offer a very rapid entry to parsing CIFs and constructing useful data structures that can be manipulated with standard Perl tools.

The use of some of the modules is illustrated in Fig. 5.3.6.1, which is a simplified version of the main program loop in the application written to typeset the CIF dictionaries printed in Part 4 of this volume.

5.3.6.1.1. *STAR::Parser*

STAR::Parser is the basic parsing module and may parse either data files or dictionaries (which may include save frames). It contains a single class method, *parse*, which returns an array of DataBlock objects. Each DataBlock object contains all the data items within an individual data block of the file. Even if the file contains only a single data block, the resulting object is passed in an array.

The contents of the data blocks may be accessed and manipulated by the methods provided by the *STAR::DataBlock* and *STAR::Dictionary* modules. They are stored internally as a multi-dimensional hash (the Perl term for an associative array with keys and associated values, which may themselves be complex data objects). Keys are provided for data blocks, save blocks, categories and data items identified during the parse. The module provides no error checking of files or objects, against the dictionary or otherwise – limited checking functionality is available through other modules in this collection.

In the example of Fig. 5.3.6.1, the *parse* method is called at line 9 to read a DDL2 dictionary file (indicated by the `-dict=>1` parameter) and return an array of data blocks. In DDL2 dictionaries (such as the mmCIF dictionary of Chapter 4.5) an entire dictionary is contained within a data block; save frames partition the data block into definitions for separate items. Normally a DDL2 CIF dictionary has only a single data block; nevertheless, the example program can handle multiple data blocks in the array, and traverses the one or several data blocks in the array through a Perl *foreach* construct (line 15).

5.3.6.1.2. *STAR::Dictionary*

The *STAR::Dictionary* module contains class and object methods for Dictionary objects created by the *STAR::Parser* module, and is in fact a subclass of *STAR::DataBlock* (see next section). Since CIF dictionaries are fully compliant STAR files, they require little that is different from the methods developed for handling data files. The method *get_save_blocks* is provided to return an array of all save frames found in the Dictionary object.

In line 25 of the example, the method is called on each dictionary loaded from the input file (as described above, normally there will only be one). The method is combined with the Perl *sort* function to create an array of save frames from the dictionary, arranged in alphabetic order. All further manipulations of the contents of these save frames will use the methods of the generic *STAR::DataBlock* class.

5.3.6.1.3. *STAR::DataBlock*

This package provides several useful methods for handling the objects within a data block returned by the *STAR::Parser* module.

The class has a constructor method *new*, which can create a completely new DataBlock object if called with no argument. This is of course essential for applications that wish to write new CIFs. Alternatively, it may be called with a `$file` argument to retrieve an existing object that has previously been written to the file system using the *store* object method described below:

```
$data_obj = STAR::DataBlock->new{ -file=>$file };
```

Table 5.3.6.1 summarizes the object methods provided by the package. The *store* method allows a DataBlock object to be serialized and written to hard disk for long-term storage. The Perl public *Storable::* module is used.

The *get_item_data* method returns the data values for a named data item. It is used frequently in the example program of Fig. 5.3.6.1; for example, at lines 28–30 the array of categories

```
 1 #!/usr/local/bin/perl
 2
 3 use STAR::Parser;
 4 use STAR::DataBlock;
 5 use STAR::Dictionary;
 6
 7 my $file = $ARGV[0];
 8
 9 @dicts = STAR::Parser->parse(-file=>$file, -dict=>1);
10
11 exit unless ( $#dicts >= 0 ); # exit if nothing there
12
13 print_document_header;
14
15 foreach $dictionary (@dicts) { # usually one
16   # Get the dictionary name and version
17   $dictname = ($dictionary->get_item_data(
18                            -item=>"_dictionary.title"))[0];
19   $dictversion = ($dictionary->get_item_data(
20                          -item=>"_dictionary.version"))[0];
21
22   print "Dictionary $dictname, version $dictversion\n";
23
24   # Get list of save-frame names, sorted alphabetically
25   @saveblocks = sort $dictionary->get_save_blocks;
26
27   foreach $saveblock (@saveblocks) { # For each save frame
28     @categories = $dictionary->get_item_data(
29                                    -save=>$saveblock,
30                                    -item=>"_category.id");
31     if ( $#categories == 0 ) { # category save frame
32       format_category($saveblock); # format the category
33
34       $category = $categories[0] ;
35       foreach $block (@saveblocks) { # re-traverse blocks
36         @itemcategoryids = $dictionary->get_item_data(
37                                    -save=>$block,
38                                -item=>"_item.category_id");
39         @itemname = $dictionary->get_item_data(
40                                    -save=>$block,
41                                    -item=>"_item.name");
42
43         if ( $#itemcategoryids == 0) { # category pointer
44           format_item($block)
45                if $itemcategoryids[0] =  /^$category$/i;
46         } elsif ( $#itemname == 0 &&
47                     $itemname[0] =  /^_$category\./i ) {
48           format_item($block);
49         }
50
51       } # end foreach of items matching current category
52     }
53   } # end foreach of save frames in the dictionary
54 } # end foreach loop per included dictionary
55
56 print_document_footer;
57 exit;
```

Fig. 5.3.6.1. Skeleton version of an application to format a CIF dictionary for publication. Only the main program fragment is shown. Line numbering is provided for referencing in the text. *format_category*, *format_item* and the *print_document_** commands are calls to external subroutines not included in this extract.

in the input dictionary is assembled by retrieving the value associated with the data item **_category.id** as the component save frames are scanned in sequence. Note that for applications within dictionaries, the method takes a `-save=>$saveblock` parameter to allow the extraction of items from specific save frames. For manipulations of data files, this parameter is omitted. In lines 17–20, the method is called without this parameter because the name and version of the dictionary are expected to be found in the outer part of the file, not within any save frame.

get_keys allows a user to display the structure of a CIF or dictionary file and can be used to analyse the content of an unknown input file. When written to a terminal, the string that is returned by this method appears as a tabulation of the items present at

Table 5.3.6.1. *Object methods provided by the STAR::DataBlock Perl module*

Method	Description
store	Saves a DataBlock object to disk
get_item_data	Returns all the data for a specified item
get_keys	Returns a string with a hierarchically formatted list of hash keys (data blocks, save blocks, categories and items) found in the data structure of the DataBlock object
get_items	Returns an array with all the items present in the DataBlock
get_categories	Returns an array with all the categories present in the DataBlock
insert_category	Inserts a category into a data structure
insert_item	Insert an item into a data structure
set_item_data	Sets the data content of an item according to a supplied array

the different levels in the data structure hierarchy, each level in the hierarchy being indicated by the amount of indentation (Fig. 5.3.6.2).

The *get_items* and *get_categories* methods are largely self-explanatory. The items or categories in the currently active DataBlock object are returned in array context.

insert_item and *insert_category* are the complements of these methods, designed to allow the insertion of new items or categories. Where appropriate (*i.e.* in dictionary applications), the save frame into which the insertion is to be made can be specified.

The remaining method, *set_item_data*, is called to set the data of item `$item` to an array of data referenced by `$dataref`:

```
$data_obj->set_item_data( -item=>$item,
                          -dataref=>$dataref );
```

As usual, an optional parameter `-save=>$save` may be included for dictionary applications where a save frame needs to be identified; the value of the variable *$save* is the save-frame name.

Note that the current version of the module does not support the creation and manipulation of data loops, although the *get_item_data* method will correctly retrieve arrays of data values from a looped list.

There are five methods available to set or retrieve attributes of a DataBlock object, namely: *file_name* for the name of the file in which the DataBlock object was found; *title* for the title of the DataBlock object (*i.e.* the name of the CIF data block with the leading **data_** string omitted); *type* for the type of data contained – 'data' for a DataBlock object but 'dictionary' for an object in the STAR::Dictionary subclass; and *starting_line* and *ending_line* for the start and end line numbers in the file where the data block is located. The method *get_attributes* returns a string containing a descriptive list of attributes of the DataBlock object.

5.3.6.1.4. *STAR::Checker*

This module implements a set of checks on a data block against a dictionary object and returns a value of '1' if the check was successful, '0' otherwise. The check tests a specific set of criteria:

(i) Are all items in the DataBlock object defined in the dictionary?

(ii) Are mandatory items present in the data block?

(iii) Are dependent items present in the data block?

(iv) Are parent items present?

(v) Do the item values conform to item type definitions in the dictionary?

Obviously, these criteria will not be appropriate for all purposes, and are in any case fully developed only for DDL2 dictionaries. An optional parameter `-options=>'1'` may be set to write a list of specific problems to the standard error output channel.

```
 data  save
block  block  categ. item
---------------------------

cif_img.dic
        -
                _category_group_list
                        _category_group_list.description
                        _category_group_list.id
                        _category_group_list.parent_id
                _dictionary
                        _dictionary.datablock_id
                        _dictionary.title
                        _dictionary.version
                _dictionary_history
                        _dictionary_history.revision
                        _dictionary_history.update
                        _dictionary_history.version
                _item_type_list
                        _item_type_list.code
                        _item_type_list.construct
                        _item_type_list.detail
                        _item_type_list.primitive_code
                _item_units_conversion
                        _item_units_conversion.factor
                        _item_units_conversion.from_code
                        _item_units_conversion.operator
                        _item_units_conversion.to_code
                _item_units_list
                        _item_units_list.code
                        _item_units_list.detail
        ARRAY_DATA
                _category
                        _category.description
                        _category.id
                        _category.mandatory_code
                _category_examples
                        _category_examples.case
                        _category_examples.detail
                _category_group
                        _category_group.id
                _category_key
                        _category_key.name
```

Fig. 5.3.6.2. Structure of the imgCIF dictionary (Chapter 4.6) as described by the *get_keys* method of the *STAR::DataBlock* module. Only the high-order file structure and the contents of the first category are included in this extract.

5.3.6.1.5. *STAR::Writer and STAR::Filter*

Two other modules are supplied by this package. *STAR::Writer* is a prototype module that can write STAR::DataBlock objects out as files in different formats; currently only the *write_cif* method exists to output a conformant CIF. *STAR::Filter* is an interactive module that prompts the user to select or reject individual categories from a STAR::Dictionary object when building a subset of the larger dictionary.

5.3.6.2. *PyCifRW*: CIF reading and writing in Python

PyCifRW (Hester, 2002) is a simple CIF input/output utility written in Python. It does not validate content against dictionaries, but it does provide a robust parser that has been extensively tested against various test files containing subtle syntactic features and against the collection of over 18 000 macromolecular CIFs available from the Protein Data Bank. The parser was implemented using the *Yapps2* parser generator (Patel, 2002) and is based on the draft Backus–Naur form (BNF) developed during a community exercise to review the CIF specification of Chapter 2.2.

As with the Perl library discussed above, *PyCifRW* presents an object-oriented set of classes and methods. Two classes are provided, *CifFile* and *CifBlock*.

A *CifFile* object provides an associative array of *CifBlock* objects, accessed by data-block name.

The methods available for the *CifFile* type are: *ReadCif(filename)*, which initializes or reinitializes a file to contain the CIF contents; *GetBlocks()*, which returns a list of the data-block names in the file; *NewBlock(blockname, [block contents])*, which adds a new data block to the file object; and *WriteOut(comment)*, which returns the contents of the current file as a CIF-conformant string, with an optional comment at the beginning. For the *NewBlock* method, the optional *block contents* must be a valid *CifBlock* object, as described below. The *NewBlock* method returns the name of the new block created, which will *not* be the requested *blockname* if a data block of the same name already exists (this conforms to the STAR and CIF requirement that data-block names must be unique within a file).

A *CifBlock* object represents the contents of a data block. The methods available to retrieve or manipulate the contents are: *GetCifItem(itemname)*, which will return the value of the data item with data name given by *itemname* (and which can be a single value or an array of looped values); *AddCifItem(data)*, which adds *data* to the current block, where *data* represents either a data name and an associated single value, or, for the case of looped data, a tuple containing an array of data names and an array of arrays of associated data values; *RemoveCifItem(dataname)*, to remove the specified data item from the current block; and *GetLoop(dataname)*, which returns a list of all data items occurring in the loop containing the data name provided. If *dataname* does not represent a looped data item, an error is returned.

The *GetLoop* method is important for the proper handling of looped data, and care is taken to handle loops robustly and efficiently. Items that are initially looped together are kept in the same loop structure.

Both *CifFile* and *CifBlock* objects act as Python mapping objects, which has the advantage that the value of a data item can be read or changed using an intuitive square-bracket notation. For example, a program can retrieve the value of a data item named `_my_data_item_name` in block 'myblockname' of a previously opened file *cf* using the following syntax:

```
value = cf["myblockname"]["_my_data_item_name"]
```

The returned value is either a single item, or a Python array of values if the data item occurs in a loop. Values are set in an analogous way and other common operations with mapping objects are also implemented.

5.3.7. Rapid development tools

The programs described so far in this chapter tend to fulfil a single purpose. Each program addresses a single clearly defined task and is of benefit to a user who has no need or desire to write a customized program. Where there is a need to write a new application, libraries of subroutines are available to the full-time programmer, such as those described in Chapters 5.4 to 5.6. However, in between these extremes, there are a large number of cases for which there is a need to combine the functionality of a number of existing programs without incurring the overhead of writing a new integrated application.

This section describes utilities that assist the development of new applications from existing software.

5.3.7.1. ZINC: an interface to CIF for standard Unix tools

Unix and derivative operating systems provide a convenient environment for rapid prototyping of programs acting on textual data. The convenience arises from the facility to chain applications together in a 'pipeline', where the output from an application may be passed directly to the input channel of another application without needing intermediate disk files for storage; and from the very rich set of utilities supplied with the typical Unix command shell, which permit files to be concatenated, split, compared, searched, stream-edited and otherwise transformed.

Many users are familiar with these utilities and can rapidly develop prototype or short-lived applications of great power by chaining them together as required. There is a temptation to use such techniques to manipulate CIFs, which as ASCII files are well suited to this. However, there are some features of the CIF syntax that are at variance with the conventional Unix idiom of storing data tags and their associated values within a basic unit of a *line* (*i.e.* a sequence of characters terminated by an end-of-line character code). Although CIFs are built from ASCII-character-populated lines, data values may be placed on a different line from that containing the parent data name (this is almost always the case for looped lists).

5.3.7.1.1. *Description of the ZINC format*

The ZINC format (Stampf, 1994) was developed to transform CIF data into an isomorphous format suitable for manipulation by standard Unix utilities.

The manner in which Unix textual utilities work suggests that an appropriate working format is one in which every individual CIF data value is available within a single-line textual record. The record must also contain information about the context of the value, conveyed through: the name of the data block in which the data value occurs; its associated data name (from which the meaning of the data value is inferred); and for recurrent data (*i.e.* values in a looped list) an indication of the list in which the data value occurs and a counter of its current occurrence in that list. In practice, each record is structured as a data line containing five TAB-separated fields in the order

blockcode name index value list-id

where *blockcode* is the name of the CIF data block (the leading `data_` string is omitted); *name* is the data name; *index* is a zero-based index of the number of occurrences of the data name within a loop (with a null value if the data occur outside a loop); *value* is the data value itself; text strings extending over several lines are collapsed into a single line with the replacement of the end-of-line character by the sequence `\n`; and *list-id* is a list identifier, stored as the data name within the list that sorts earliest (because it is a purely syntactic transformation, the utility does not consult a dictionary file for the correct `_list_reference` identifying token).

Comments in the CIF are also stored, to permit regeneration of the original file by an inverse transformation; and because it is often convenient to read such interpolations, especially in interactive activities with CIFs of which the user has no prior knowledge. Comments are stored with a value of '`(`' in the data-name field. They are also numbered (starting from zero) in the index field of the ZINC record.

Most details of the ZINC transformation are illustrated by the simple example in CIF format shown in Fig. 5.3.7.1(*a*). This file transformed to ZINC format would appear on a display terminal as illustrated in Fig. 5.3.7.1(*b*). However, the spacing is deceptive, since typical display terminals convert TAB characters to a variable number of spaces. A more accurate (albeit less legible) representation of the output is given in Fig. 5.3.7.1(*c*).

```
# A simple CIF

data_object

# description of a simple
# polygon
  _name
;
triangle
;
  loop_
      _x _y
      0.0       0.0
      1.0       0.0
      0.0       1.0

      _num_sides 3
```

(*a*)

```
       (        0        # A simple CIF
object (        1        # description of a simple
object (        2        # polygon
object _name             ;\ntriangle\n;
object _x       0        0.0     _x
object _y       0        0.0     _x
object _x       1        1.0     _x
object _y       1        0.0     _x
object _x       2        0.0     _x
object _y       2        1.0     _x
object _num_sides        3
```

(*b*)

```
␣(␣0␣# A simple CIF␣
object␣(␣1␣# description of a simple␣
object␣(␣2␣# polygon␣
object␣␣name␣␣;\ntriangle\n;␣
object␣␣x␣0␣0.0␣␣x
object␣␣y␣0␣0.0␣␣x
object␣␣x␣1␣1.0␣␣x
object␣␣y␣1␣0.0␣␣x
object␣␣x␣2␣0.0␣␣x
object␣␣y␣2␣1.0␣␣x
object␣␣num_sides␣␣␣3␣
```

(*c*)

Fig. 5.3.7.1. ZINC transformation of CIF. (*a*) is a sample file in CIF format. (*b*) shows the output to a display terminal when this file is transformed by *cifZinc*. (*c*) The same output as (*b*), but with TAB characters represented by special graphical characters.

Note the following points: the data-block name is initially null; each comment line is numbered in sequence from zero; the index field is null for data names that are not within looped lists.

5.3.7.1.2. *ZINC-based utilities*

The purpose of ZINC is to form an intermediate stage in pipeline processes involving Unix tools, and therefore CIF data in ZINC format have only a transitory existence. The ZINC distribution package provides the complementary tools *cifZinc* and *zincCif* required to interconvert formats; and in addition a few sample applications are provided as examples for Unix programmers.

5.3.7.1.2.1. *cifZinc*

cifZinc takes a CIF name as a command-line argument or the CIF itself from standard input and produces a ZINC-format file on standard output. It has one option, '*-c*', which removes comments (which arguably have no place in a CIF).

5.3.7.1.2.2. *zincCif*

zincCif is a Perl script that takes a ZINC-format file (again from standard input or as a name on the command line) and pretty prints the corresponding CIF to standard output. Often, the pipeline `cifZinc a.cif | zincCif > b.cif` produces a more attractive CIF than the original.

5.3.7.1.2.3. *zincGrep*

The shell script *zincGrep* is the utility most requested by those seeing a CIF for the first time. It allows a regular-expression search of a ZINC-format file (or a CIF specified on the command line, which is converted to a ZINC-format file first) and reports the block name, data name, index and value. For example, if the file describing a triangle in the last section were called simple.cif, the command `zincGrep _name simple.cif` would produce

```
object _name            ;\ntriangle\n;
```

5.3.7.1.2.4. *cifdiff*

cifdiff is a C-shell script that takes two CIFs and lists the differences between them. Unlike the standard Unix utility *diff*, which compares files line-by-line, *cifdiff* can determine differences that are independent of reordering and white-space padding.

This script takes each CIF, converts it to a ZINC-format file, then sorts it, first based on the data-block name, then (keeping the loops together) on the data name. It then removes the last field (which is not part of the CIF) and stores the remainder in temporary files. It then runs the standard *diff* program against these reordered temporary files. This is remarkably effective both in finding any differences and in providing the context (it names the block and data name as well as the value) needed to understand the differences.

Fig. 5.3.7.2 illustrates two CIFs that are very different in the presentation of their contents, but have only a small difference of substance in their content. Fig. 5.3.7.3 indicates the output from *cifdiff* that identifies the changed data.

```
data_sample
     _title  'scatter graph'
     _description
; A collection of x, y coordinates of points
  drawn in specified colours
;
     loop_
         _x  _y  _colour
          0   0   red
          1   1   red
          2   4   red
          3   9   orange
          4   16  orange
          5   25  orange
      _status complete
```

(*a*)

```
data_sample
_status complete
loop_    _y  _x  _colour
          0   0   red
          1   1   red
          2   2   red
          9   3   orange
          16  4   orange
          25  5   orange
_title  'scatter graph' _description
; A collection of x, y coordinates of points
  drawn in specified colours
;
```

(*b*)

Fig. 5.3.7.2. Two example files in CIF format differing greatly in layout but little in content: (*a*) sample1.cif, (*b*) sample2.cif.

```
% cifdiff sample1.cif sample2.cif
18c18
< sample         _y      2       4
---
> sample         _y      2       2
```

Fig. 5.3.7.3. Output from *cifdiff* comparing files sample1.cif and sample2.cif of Fig. 5.3.7.2. The entries in each line comprise the data-block name, the variable name, the zero-based index of the occurrence of the value in a looped list and the value. Hence it is the *third* value of **_y** in the loop that has changed.

5.3.7.1.2.5. *zb*

zb is a small (less than 200 lines) Tcl/Tk program (Ousterhout, 1994) that provides a simple graphical front end to a ZINC-format file or CIF allowing the user to browse through the contents. Multiple files can be viewed simultaneously, as can multiple data blocks, on any X terminal. *zb* recognizes command-line argument file names in the form *.cif as being in CIF format and converts them to ZINC format automatically.

5.3.7.1.2.6. *zincNl*

zincNl is a Perl script that takes a ZINC file and creates a Fortran-compatible namelist file allowing for easy access to any CIF by Fortran programs without the need for extensive I/O libraries or reprogramming. As with *zb* above, it will automatically convert a CIF to a ZINC-format file if it needs to.

For a more substantial tool providing CIF input functions in Fortran and C, see the discussion below of *CifSieve* (Section 5.3.7.2).

5.3.7.1.2.7. *zincSubset*

zincSubset is another C-shell script which is very short but very useful. It allows a user to generate a custom subset of any ZINC-format file (or CIF) simply by listing the desired data blocks and data names. The script has two file arguments, the first of which specifies a file with regular expressions that specify what is to be included in the subset, and the second of which is the ZINC-format file itself (or standard input). It allows two options: '*-c*' to remove comments and '*-v*' to invert the sense of the search.

It converts the CIF into a ZINC-format file, uses the Unix *grep* program to search through the ZINC-format file for patterns that appear in the regular-expression file and pretty prints the result. For example, the command

```
zincSubset defs cif_core.dic > cifdic.defs
```

will produce a subset of the core CIF dictionary that contains only the names and definitions when the file named 'defs' contains two lines with the TAB-surrounded word '`_name`' and the TAB-surrounded word '`_definition`'.

All the tools listed in this section operate by design in concert with each other, providing the opportunity for generating increasingly complex tools. For example, to generate the Fortran namelist input file with only certain data items, a pipeline of *zincSubset* and *zincNl* will suffice. As more tools are developed, the range of applications will increase many-fold.

5.3.7.2. *CifSieve*: automatic construction of CIF input functions

Among the utilities described in the ZINC package above was a tool to generate Fortran namelist files. It is a common requirement of applications developers that they should be able swiftly to convert existing programs to read CIF data. While libraries such as *CIFtbx* (Chapter 5.4) and *CIFLIB* (Westbrook *et al.*, 1997) offer very powerful functions for building CIF applications, it can be time-consuming to integrate them with existing software. It is a goal of *CifSieve* (Hester & Okamura, 1998) to enable the rapid creation of new CIF-conversant software by using a CIF dictionary as a template for input data structures.

The *CifSieve* program runs on Unix systems with installed versions of the software utilities and programming languages *bison* or *yacc*, *flex*, Perl (Wall *et al.*, 2000) and C.

5.3.7.2.1. *Overview of the process*

The data names in a CIF are defined in a dictionary written in DDL1 or DDL2 formalism. Therefore, information about the data type and array structure of data variables is already to hand for a software author wishing to determine how to read CIF data into a program's data structures. The *CifSieve* process requires that the programmer augment the relevant CIF dictionary by adding to a copy of the definition of desired items a new attribute, named `_variable_name`, that passes to the application program the name of the associated program variable.

A program *BuildSiv* then reads the augmented dictionary and produces a subroutine capable of reading a CIF and transferring the data items tagged in the augmented dictionary to internal variable storage. The associated data structure is presented in an ancillary file which must be linked to the application program.

CifSieve can produce input subroutines and header or include files for C and Fortran language programs. For C applications, the input subroutine is called *cifsiv_* and is invoked with arguments *cifsiv_(CIF, block)* where *CIF* is the name of the input CIF and *block* is the name of the data block from which data should be read. The data structure is declared in a header file cifvars.h which must be included in subroutines that manipulate the data input from the CIF. For Fortran applications, the input subroutine is also called *cifsiv_*, but takes an additional argument, *blockbeg*, which is the address of the common block containing the input variable names, declared in the include file forcif.inc.

5.3.7.2.2. *The augmented DDL dictionary*

Fig. 5.3.7.4 is an example of the annotations necessary to flag the data names that refer to data items desired to be input from a CIF. The current implementation requires that a copy of the DDL dictionary relevant for the CIF be physically edited to include the new `_variable_name` attribute. The inclusion of such a new attribute will not affect the use of the CIF dictionary for other purposes and by other software.

The definition blocks of data items that are not to be read by the application should be left unchanged.

The value assigned to the `_variable_name` attribute is the name of the variable declared in the application program for storing the input data item. If the items to be input are part of an array (*i.e.* they exist in the CIF as a looped list), the variable name should be supplied as a dimensioned array variable, *e.g.* `atsiteu[1000]` in the example of Fig. 5.3.7.4.

The same attribute (`_variable_name`) may be inserted in DDL1 or DDL2 dictionaries. Separate parsers are supplied for use with either format. When *BuildSiv* is invoked, the parser reads the augmented dictionary and identifies the data items required by the target input subroutine by the presence of a `_variable_name` attribute in the definition block. The definition is read and the relevant values of the type (DDL attribute `_type`), item name (`_name`) and variable name are output in a simple tag–value format and in a standard order. For DDL2 dictionaries, values of `_item_aliases.alias_name` and `_item_linked.parent_name`, if present, are also output. The DDL parser thus transforms and simplifies the dictionary contents.

Where the item-name attribute occurs inside a loop (*i.e.* several data names occur in a single definition block in the dictionary), the variable name for that particular definition block will be given an extra array dimension by *CifSieve*, equal to the number of names in the loop. When a name from this loop is found in a CIF, the value will be read into the respective array location. If an `_item_aliases.alias_name` attribute is present (DDL2), the alias will also be recognized in CIF input files. If this attribute occurs together with looped item names in the domain dictionary, an attempt is made to determine the parent `_item.name` in the loop to which this `_item_aliases.alias_name` refers. This is done within the *BuildSiv* program by examining `_item_linked.parent_name` entries within the same definition block.

Data typing is simplified; the `_item_type.code` values of DDL2 dictionaries are collapsed onto primitive 'numb' or 'char' types. Values of type numb are declared and stored as type double (C) or REAL*8 (Fortran), while values of type char are stored as character arrays char[84] (C) or CHARACTER*84 (Fortran). In consequence, multiple lines of text *cannot* be retrieved with this version of *CifSieve*. Note in particular that values declared as of type 'int' in DDL2 dictionaries will be stored as double-precision real.

```
data_atom_site_aniso_label
    _name                 '_atom_site_aniso_label'
    _category               atom_site
    _type                   char
    _variable_name          mylabel[50]
    _list                   yes

data_atom_site_aniso_U_
    loop_ _name           '_atom_site_aniso_U_11'
                          '_atom_site_aniso_U_12'
                          '_atom_site_aniso_U_13'
                          '_atom_site_aniso_U_22'
                          '_atom_site_aniso_U_23'
                          '_atom_site_aniso_U_33'
    _category               atom_site
    _variable_name          atsiteu[1000]
    _type                   numb

data_reflns_number_
    loop_ _name           '_reflns_number_total'
                          '_reflns_number_observed'
    _category               reflns
    _type                   numb
    _enumeration_range      0:
    _variable_name          reftot

data_refine_ls_extinction_method
    _variable_name          extmet
    _name                 '_refine_ls_extinction_method'
    _category               refine
    _type                   char
    _enumeration_default  'Zachariasen'
```

Fig. 5.3.7.4. Extracts from an augmented DDL1 dictionary (version 1.0 of the core CIF dictionary). The additional `_variable_name` entry is shown in italics.

```
/* These declarations have been automatically
   generated by the cif file input/output function
   generator.  This file should be included in any
   routines that call these functions */
typedef char cifstring[84];
           /* to avoid array complications later */
#define MYLABELMAX 50
#define ATSITEUMAX  1000
typedef double atsiteutype [6];
#ifdef CIFVARDEC
  cifstring errormes;         /* an error message */
  int       errornum;         /* an error number  */
  cifstring mylabel[50];
                     /*data_atom_site_aniso_label*/
  atsiteutype atsiteu[1000];
                        /*data_atom_site_aniso_U_*/
  atsiteutype atsiteuesd[1000];
  cifstring extmet;
               /*data_refine_ls_extinction_method*/
  double reftot [2];         /*data_reflns_number_*/
  double reftotesd [2];
#else
  extern cifstring errormes;  /* an error message */
  extern int       errornum;  /* an error number  */
  extern cifstring mylabel[50];
                     /*data_atom_site_aniso_label*/
  extern atsiteutype atsiteu[1000];
                        /*data_atom_site_aniso_U_*/
  extern atsiteutype atsiteuesd[1000];
  extern cifstring extmet;
               /*data_refine_ls_extinction_method*/
  extern double reftot [2];  /*data_reflns_number_*/
  extern double reftotesd [2];
#endif
```

Fig. 5.3.7.5. Header file cifvars.h for a C application built by *BuildSiv* from the augmented DDL dictionary of Fig. 5.3.7.4.

```
/* A simple example application of the automatically
   generated cifsiv_ function */

#include <stdio.h>
#include "cifvars.h"

main(int argc, char *argv[])
{
  int i;
  char filename[80];
  char block[80];
  printf("Please enter CIF file name: ");
  scanf("%s", filename);
  printf("Please enter data block name ");
  printf("(without data_ prepended): ");
  scanf("%s", block);
  errornum = 0;
  cifsiv_(filename,block);
  if(errornum != 0)  /* an error, we have problems */
    {
    printf("An error occurred in reading the
CIF:\n");
    printf("%s",errormes);
    }
  for(i=0;i<5;i++)
  {
    printf("Atom %d: %s %f %f\n", i, mylabel[i],
            atsiteu[i][0], atsiteu[i][1]);
  }
  printf("Total reflections: %.f\n", reftot[0]);
  printf("Extinction method: %s\n", extmet);
}
```

Fig. 5.3.7.6. An example C program designed to read CIF data as tagged in the augmented DDL dictionary of Fig. 5.3.7.4.

5.3.7.2.3. *Input to a C application program*

When a DDL dictionary *dictfile* has been edited in accordance with the description above, the program *BuildSiv* may be run under a Unix-like operating system with a command of the form

`BuildSiv` *`dictfile ddlversion`*

where *ddlversion* takes the values '1' or '2' to indicate that a DDL1 or DDL2 parser is appropriate. If the option '*-e*' is given before *dictfile*, variable definitions and read capability for standard uncertainty values will be included as well. The name of the variable that will hold the standard uncertainty is the name given by the programmer with the string `esd` appended.

An object file cifsiv.o is produced together with a header file cifvars.h. Some source-code files are also produced as intermediate files in the lexical analysis and parse phases of the software build; these may be deleted. The object file must be linked against the other object files when the application program is compiled and references to the header files must be introduced (generally through C preprocessor `#include` directives) within the application code where access to the imported data structures is required.

Fig. 5.3.7.5 is an example of the header file cifvars.h built when *BuildSiv* reads the augmented dictionary of Fig. 5.3.7.4 with the '*-e*' option to interpret and store standard uncertainties.

The integer variable *errornum* stores a nonzero value if an error occurs in attempting to read a CIF, and an error message is stored in the character array *errormes*, indicating the nature of the problem. Errors generated by the input subroutine *cifsiv_* are not fatal to the parent application program, and will at worst discard the particular loop block or data item affected. The parser operates by discarding CIF data upon encountering an error until it reaches an understandable set of input values. So, for example, if three numbers appear after an item name instead of one, the second two will be ignored after the error variables have been set, and parsing will continue. Similarly, if a serious error occurs within a loop, such as the appearance of an item name not matching an array variable, the entire loop is normally ignored. If a new packet of looped data exceeds the specified array limits, all further data in that loop are ignored.

The *cifsiv_* function has prototype

`void` *`cifsiv_`*`(char*` *`filename`*`, char*` *`blockname`*`)`

and requires pointers to character strings containing the name of the input file and the data-block code from which input is required.

A simple example C application illustrating the use of the *cifsiv_* subroutine is given in Fig. 5.3.7.6.

5.3.7.2.4. *Input to a Fortran application program*

A Fortran program can make use of the C input function generated by *BuildSiv* as long as the compiler used is capable of linking C and Fortran modules. For Fortran applications, the '*-f*' command-line option is used:

`BuildSiv -f` *`dictfile ddlversion`*

A C structure is defined for use within the *cifsiv_* subroutine and an identically constructed Fortran common block is built for use within Fortran routines. The first variable within the common block *must* be passed as an additional argument when the *cifsiv_* function is called. In the current implementation, that variable is

```
C The following common block corresponds to a
C structure defined in the C header, which is written
C to by routine 'cifsiv'. In order to correctly write
C to this common block, 'cifsiv' should be called
C with a *third* argument which will always be
C 'blockbeg'.
     REAL BLOCKBEG
     CHARACTER*84  ERRORMES
     INTEGER ERRORNUM
     CHARACTER*84 mylabel(50)
     REAL*8 atrat(50)
     REAL*8 atratesd(50)
     REAL*8 atsiteu(6,500)
     REAL*8 atsiteuesd(6,500)
     CHARACTER*84 extmet
     REAL*8 reftot(2)
     REAL*8 reftotesd(2)
     COMMON/CIFCMN/BLOCKBEG,ERRORMES,ERRORNUM,mylabel,
    *atrat,atratesd,atsiteu,atsiteuesd,extmet,reftot,
    *reftotesd
```

Fig. 5.3.7.7. Fortran include file forcif.inc for an application built by *BuildSiv* from the augmented DDL dictionary of Fig. 5.3.7.4.

```
      PROGRAM FORGET
      include 'forcif.inc'
      call cifsiv("tbshort.cif","tbal03",blockbeg)
      do i = 1,4
        write(*,*) mylabel(i), atsiteu(1,i),
     *             atsiteu(2,i)
      enddo
      write(*,*) reftot(1)
      write(*,*) extmet
      end
```

Fig. 5.3.7.8. An example Fortran program designed to read CIF data as tagged in the augmented DDL dictionary of Fig. 5.3.7.4.

always called 'BLOCKBEG'. The input subroutine is thus called from within a Fortran program by a line of the type

`CALL CIFSIV(`*FILE*`,` *BLOCK*`,` `BLOCKBEG)`

where *FILE* and *BLOCK* are, respectively, the name of the input file and data block.

Fig. 5.3.7.7 is an example Fortran include file generated by *BuildSiv* and Fig. 5.3.7.8 is an example application incorporating this file. As with the C examples, the CIF data to be read are those specified in the dictionary augmented according to Fig. 5.3.7.4.

It may be noted that the C header file generated by the Fortran implementation of *BuildSiv* (and which is used directly by the C object file produced) is callable by any other C program or subroutine. The Fortran common block is represented by a C structure named *cifcmnptr*, so that the variable names are stored within that structure and must be addressed through the C → operator. That is, an additional C routine compiled in with the Fortran example program of Fig. 5.3.7.7 would refer to the variable holding the value of the input `_refine_ls_extinction_method` as `(char *)cifcmnptr->extmet`.

5.3.8. Tools for mmCIF

The complex relationships between the components of a macromolecular structure at various levels of detail are richly described by the data names in the mmCIF dictionary, but their number and complexity demand more heavyweight tools for proper handling. Input/output for small-molecule or inorganic structures can often be handled by a simple CIF parse and identification of the desired components of one or a few looped data structures. For macromolecules, multiple categories must be loaded simultaneously, and the integrity of relationships between items in the different categories must be properly maintained. For this reason, the most effective tools for mmCIF-based applications have high-level interactions with the mmCIF or related dictionaries, and necessarily involve more complex data manipulations.

In this section are discussed three software systems that are available for work with macromolecular structures: *CIFOBJ* and related libraries, which provide a long-established and complete application program interface (API) to dictionaries and data files; *OpenMMS*, an exciting development allowing abstract data representations (based on the mmCIF dictionary definitions) to be exchanged between applications using an intermediate middleware layer; and *mmLib*, which is a Python toolkit for biomolecular structure applications. These latter two may come closer to the area of domain-specific applications than most of the generic tools we have discussed in this chapter. However, they demonstrate how the abstract data model represented by the mmCIF dictionaries can effectively be imported into a diverse range of programming environments.

5.3.8.1. *CIFOBJ* and related libraries

Early in the development of the mmCIF dictionary, the Nucleic Acid Database at Rutgers University (Berman *et al.*, 1992) created a number of CIF libraries and utilities to underpin data-processing activities. Much of this development work was carried across when the curatorship of the Protein Data Bank was transferred to the Research Collaboratory for Structural Bioinformatics (RCSB; Berman *et al.*, 2002), and the software provides the engine for many of the robust and industrial-strength database operations of these organizations.

CIFLIB (Westbrook *et al.*, 1997) was an early class library, no longer supported, that was developed to provide an API to macromolecular CIF data files and to the associated dictionaries (Chapters 3.6 and 4.5) and underlying dictionary definition language (DDL2) files (Chapter 2.6).

The RCSB Protein Data Bank now distributes object-oriented parsing tools (*CIFPARSE_OBJ*; Tosic & Westbrook, 2000) which fully support CIF data files and their underlying metadata descriptions in dictionaries and DDL2 attribute sets, and a comprehensive library of access methods for data and dictionary objects at category and item level.

The information infrastructure of the Protein Data Bank, built upon these tools, is discussed in Chapter 5.5. All the software produced for this purpose is distributed with full source under an open-source licence, to promote the development of mmCIF tools and to encourage interoperability with other software environments.

5.3.8.2. *OpenMMS*

Object classes represent the first stage in abstracting related data components. By building structured software modules that can manage the small-scale interactions between data components, the programmer can write more succinct code to handle the interactions between much higher-level data constructs. An API then permits third parties to handle the larger-scale objects without any need to know the internal workings of the class library. The next logical step is to present a standard set of 'objects' representing complete logical entities to any programmer for 'plug-and-play' incorporation into new applications.

The Life Sciences Research domain task force of the Object Management Group (OMG, 2001) is concerned with the development of standards for data exchange in biomolecular sciences, and in 2002 approved a macromolecular structure Corba specification. Corba (the common object request broker architecture) is a middleware architecture intended to serve just this purpose of providing access to standard objects representing discrete logical entities suitable for programmatic manipulation. Corba promotes interoperability across networked applications by separating entirely the API from the implementation of the underlying data objects. For applications such as the macromolecular structures database hosted by the Protein Data Bank, the attraction of networked interoperability is that information can be accessed through distributed and federated databases, and can be delivered on demand to any compatible software.

A Corba application comprises an interface definition language (IDL) and an API that together define access to a data structure that encapsulates the abstract representation of the objects and relationships relevant to a particular area of knowledge. In general terms, this data structure may be described as an 'ontology' (Westbrook & Bourne, 2000). The ontology adopted for macromolecular structure (MMS) data was based on the mmCIF dictionary following a submission by the Research Collaboratory for Structural Bioinformatics to a Request for Proposal (Greer, 2000).

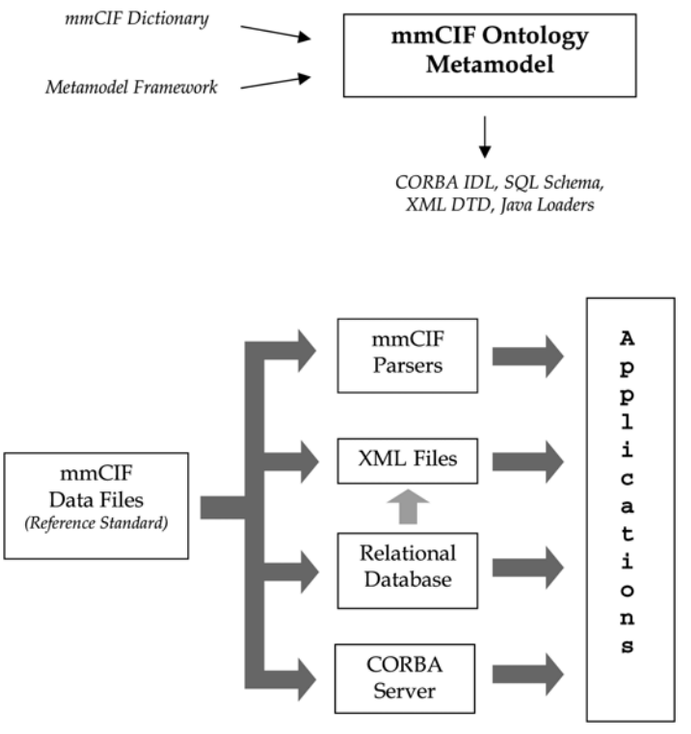


Fig. 5.3.8.1. The *OpenMMS* metamodel and data flow.

5.3.8.2.1. *The OpenMMS toolkit*

In practice, the ontology was developed in a 'metamodel' that combined the definitions and relationships between data items specified in the mmCIF dictionary with a generic metamodel framework. The metamodel extracts the information in the mmCIF dictionary but maintains it in a representation that is independent of the mmCIF STAR or any other file format. The standard building block of the metamodel is an *Entry* object, modelling a single macromolecular structure.

From a suitable metamodel, it then becomes relatively straightforward to generate alternative expressions of the information to suit different access requirements. The *OpenMMS* toolkit (Greer *et al.*, 2002) was built using Java source code to generate a Corba interface, an SQL schema for relational database loading and an XML representation of macromolecular data sets (Fig. 5.3.8.1).

The toolkit contains an mmCIF parsing module capable of direct access to the underlying data archive of mmCIF data files. This is important, because the data files represent a common reference for all the derived representations. Any errors or discrepancies between the expressed forms of the Corba, XML or SQL representations are resolved against the standard mmCIF reference form.

The relational database supporting an SQL-92 compatible interface provides an appropriate API for many applications, particularly ones that require extensive string searches. The close relationship between the mmCIF data model and relational database models has already been described earlier in this volume (Chapter 2.6).

Advantages of the SQL interface are that it provides rapid access direct to the binary data storage representation and that individual components of a data set may be efficiently retrieved without the need to search sequentially through an entire entry.

This efficiency of access and the ability to retrieve individual MMS data elements from a remote server is best realized through the Corba interface, the primary purpose of which is indeed to facilitate such high-performance access.

The bulk exchange of data is addressed through the generation of XML files. XML is a simple, powerful and widely used standard for interchanging data, and its use for transporting macromolecular data obviates the need for target applications to build their own STAR parsers. However, the use of markup tags around every individual data element does make the files much larger than their mmCIF progenitors. This is not an insurmountable problem in large-scale application environments, but it can undermine the effectiveness of XML as a representation mechanism in such applications as web browsers. A possible approach to this could be to define different, less verbose, XML representations and populate these on demand from a database store, either by SQL or XML queries. This is not an approach that the current *OpenMMS* toolkit supports directly.

Fig. 5.3.8.2 is an extract from an XML data file generated from the PDB structure 1xy2. The XML uses a reserved name space *PDBx* conforming to the schema http://deposit.pdb.org/pdbML/pdbx-v0.905.xsd. Data tags map cleanly to the corresponding data names in the mmCIF dictionary formed by concatenating the XML element name with its parent category name. For example, the entry `<PDBx:length_a>27.080</PDBx:length_a>` included in the `<PDBx:cellCategory>` container tag can be directly translated to the corresponding mmCIF data item **`_cell.length_a 27.080`**. CIF data loops are represented by repeated instances of the XML tag representing the corresponding CIF data name (for example, the multiple `<PDBx:audit_author name>` tags are equivalent to a CIF **`loop_ _audit_author.name`** construct). Nonstandard items with a *pdbx_* prefix (*e.g.* `<PDBx:pdbx_description>` in the `<PDBx:entityCategory>` group) refer to private data names in the PDB extension dictionary (Appendix 3.6.2).

5.3.8.3. *mmLib*: a Python toolkit for bioinformatics applications

While the libraries developed for use within the Protein Data Bank provide powerful functionality, their very size and complexity make them inappropriate for some applications. Indeed, considerable effort may be needed to compile the C++ code on non-standard platforms. The *mmLib* toolkit (Painter & Merritt, 2004)

```
<?xml version="1.0" encoding="UTF-8" ?>
<PDBx:datablock  datablockName="1XY2"
 xmlns:PDBx="http://deposit.pdb.org/pdbML/pdbx-v0.905.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
    "http://deposit.pdb.org/pdbML/pdbx-v0.905.xsd
         pdbx-v0.905.xsd">
 <PDBx:audit_authorCategory>
  <PDBx:audit_author name="Cooper, S."></PDBx:audit_author>
  <PDBx:audit_author name="Blundell, T.L.">
                                      </PDBx:audit_author>
  <PDBx:audit_author name="Pitts, J.E."></PDBx:audit_author>
  <PDBx:audit_author name="Wood, S.P."></PDBx:audit_author>
  <PDBx:audit_author name="Tickle, I.J."></PDBx:audit_author>
 </PDBx:audit_authorCategory>
 <PDBx:cellCategory>
  <PDBx:cell entry_id="1XY2">
   <PDBx:length_a>27.080</PDBx:length_a>
   <PDBx:length_b>9.060</PDBx:length_b>
   <PDBx:length_c>22.980</PDBx:length_c>
   <PDBx:angle_alpha>90.00</PDBx:angle_alpha>
   <PDBx:angle_beta>102.06</PDBx:angle_beta>
   <PDBx:angle_gamma>90.00</PDBx:angle_gamma>
   <PDBx:Z_PDB>4</PDBx:Z_PDB>
  </PDBx:cell>
 </PDBx:cellCategory>
 <PDBx:citationCategory>
  <PDBx:citation id="primary">
   <PDBx:title>Crystal structure analysis of
               deamino-oxytocin: conformational flexibility
               and receptor binding.</PDBx:title>
   <PDBx:journal_abbrev>Science</PDBx:journal_abbrev>
   <PDBx:journal_volume>232</PDBx:journal_volume>
   <PDBx:page_first>633</PDBx:page_first>
   <PDBx:page_last>636</PDBx:page_last>
   <PDBx:year>1986</PDBx:year>
   <PDBx:journal_id_ASTM>SCIEAS</PDBx:journal_id_ASTM>
   <PDBx:country>US</PDBx:country>
   <PDBx:journal_id_ISSN>0036-8075</PDBx:journal_id_ISSN>
   <PDBx:journal_id_CSD>0038</PDBx:journal_id_CSD>
  </PDBx:citation>
 </PDBx:citationCategory>
 <PDBx:computingCategory>
  <PDBx:computing entry_id="1XY2">
   <PDBx:structure_solution>SHELX</PDBx:structure_solution>
   <PDBx:structure_refinement>SHELX-76
                                </PDBx:structure_refinement>
  </PDBx:computing>
 </PDBx:computingCategory>
 <PDBx:database_2Category>
  <PDBx:database_2 database_id="PDB" database_code="1XY2">
                                            </PDBx:database_2>
 </PDBx:database_2Category>
 <PDBx:entityCategory>
  <PDBx:entity id="1">
   <PDBx:type>polymer</PDBx:type>
   <PDBx:src_method>man</PDBx:src_method>
   <PDBx:pdbx_description>OXYTOCIN</PDBx:pdbx_description>
   <PDBx:formula_weight>978.189</PDBx:formula_weight>
   <PDBx:pdbx_number_of_molecules>1
                             </PDBx:pdbx_number_of_molecules>
  </PDBx:entity>
  <PDBx:entity id="2">
   <PDBx:type>water</PDBx:type>
   <PDBx:src_method>nat</PDBx:src_method>
   <PDBx:pdbx_description>water</PDBx:pdbx_description>
   <PDBx:formula_weight>18.015</PDBx:formula_weight>
   <PDBx:pdbx_number_of_molecules>7
                             </PDBx:pdbx_number_of_molecules>
  </PDBx:entity>
 </PDBx:entityCategory>
```

Fig. 5.3.8.2. Sample XML output from the *OpenMMS* XML generator. Lines have been omitted or wrapped to fit the present column width.

addresses this by supplying a library of object-oriented routines implemented in Python (van Rossum, 1991) that are designed to integrate with existing or new applications in an easy way.

The objective of *mmLib* is to build a support platform to handle the increasingly rich data about macromolecular structure

Table 5.3.8.1. *The modules provided by the mmLib toolkit*

mmLib.mmCIF	mmCIF parser
mmLib.PDB	PDB format parser
mmLib.Library	Base chemical library
mmLib.Extensions.CCP4Library	Data retrieval from CCP4 monomer library
mmLib.Elements	Chemical data for elements
mmLib.AminoAcids	Chemical data for amino acids
mmLib.NucleicAcids	Chemical data for nucleic acids
mmLib.Structure	Macromolecular structure model
mmLib.GLViewer	OpenGL visualizer

```
import mmLib
from mmLib.FileLoader import LoadStructure, SaveStructure
    struct = LoadStructure(
                fil = cif,
                format = "PDB",
                build_properties = ("no_bonds",) )
    SaveStructure(
        fil = pdb,
        structure = struct,
        format = "CIF")
```

Fig. 5.3.8.3. A snippet of code illustrating mmCIF/PDB file format conversion with the *mmLib* toolkit.

available to structural biologists. Not only do applications need to be able to handle atomic positions and build appropriate three-dimensional structure representations; but links to and integration with information on sequence, homologous structures, and biochemical, genetic and medical form and function are also demanded from individual program systems. Since much of these data are available from external databases in a variety of formats, *mmLib* will not be restricted to the handling of files in a single format. Its initial release provides support for mmCIF, for the PDB format files that historically have been used for representation of macromolecular structures (Westbrook & Fitzgerald, 2003) and for the MTZ format used by the *CCP4* program suite (Collaborative Computational Project, Number 4, 1994).

Table 5.3.8.1 lists the main modules in the current release. *mmLib.mmCIF* and *mmLib.PDB* are read/write parsers for mmCIF and PDB format files, respectively, which handle file input and output in these formats, and provide support for inspection or modification of such file formats. They are typically used in conjunction with the *mmLib.FileLoader* component to populate the *mmLib.Structure* internal representation of the macromolecular structure. The high-level abstraction of such functionality allows for very succinct programmatic constructs. Fig. 5.3.8.3 illustrates this with a program snippet that (apart from the necessary system calls for file management) achieves the conversion of an mmCIF input file to a PDB format representation. This is sufficiently robust and lightweight to act as an input filter to software already designed for handling PDB format files.

mmLib.Structure represents the internal representation of a molecular structure and is implemented as an object hierarchy with four basic object classes: *Structure*, *Chain*, *Fragment* and *Atom*. The *Fragment* class has subclasses *AminoAcidResidue* and *NucleicAcidResidue*. In order to build a complete representation of a structure, the toolkit may need to load data from an input mmCIF or PDB format file, and also from standard data sets of properties of individual monomers and chemical elements; these standard libraries of chemical properties are provided by the *mmLib.Library* module. The core *mmLib* source includes a limited library of such chemical properties (accessible through the subclasses *mmLib.Elements*, *mmLib.AminoAcids* and *mmLib.NucleicAcids*)

and also provides support for the extensive CCP4 monomer library through the *mmLib.Extensions.CCP4Library*. The naming of this class expresses the intention that other standard data sources should be made accessible in the same way.

The CCP4 monomer library is in fact included with the software as a directory tree of small files in mmCIF format, which are loaded into the *Structure* object through the normal use of the toolkit's mmCIF parser.

mmLib.GLViewer is a module provided to support visualization programs using the OpenGL graphics environment. Although it does not by itself provide a stand-alone viewer, it can be incorporated into many common graphics application building environments. An example molecular viewer, *mmView*, is provided with the distribution as an example of an application using the GTK graphical user interface, a popular toolkit in Linux.

5.3.9. Concluding remarks

CIF is a domain-specific format that cannot attract the number of programmers that generic formats such as XML do. In spite of this, there is an impressive collection of programs available to support activities at many levels, from the single-line shell script needed to search for some desired content in a collection of CIFs, to the industrial-scale activities of major databases and publishing houses. As many examples as possible of the programs discussed in this chapter have been collected on the CD-ROM accompanying this volume and on the IUCr web site (http://www.iucr.org/iucr-top/cif/software). It is hoped that the contributions described here will inspire future generations of programmers to contribute to a growing and increasingly robust software collection to make the use of CIFs ever easier and more fruitful.

I am immensely grateful for the assistance, cooperation and involvement of the community of software authors who have contributed to this chapter in one way or another, and to all the programmers and developers who have been active through the cif-developers discussion list of the IUCr (http://www.iucr.org/iucr-top/lists/cif-developers) and in private discussions.

References

Allen, F. H. (2002). *The Cambridge Structural Database: a quarter of a million crystal structures and rising*. *Acta Cryst.* B**58**, 380–388.

Allen, F. H., Johnson, O., Shields, G. P., Smith, B. R. & Towler, M. (2004). *CIF applications. XV. enCIFer: a program for viewing, editing and visualizing CIFs. J. Appl. Cryst.* **37**, 335–338.

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D. & Zardecki, C. (2002). *The Protein Data Bank. Acta Cryst.* D**58**, 899–907.

Berman, H. M., Olson, W. K., Beveridge, D. L., Westbrook, J., Gelbin, A., Demeny, T., Hsieh, S.-H., Srinivasan, A. R. & Schneider, B. (1992). *The Nucleic Acid Database: a comprehensive relational database of three-dimensional structures of nucleic acids. Biophys. J.* **63**, 751–759.

Bernstein, H. J. (1998). *cif2cif. CIF copy program.* http://www.iucr.org/iucr-top/cif/software/ciftbx/cif 2cif.src/.

Bernstein, H. J. & Hall, S. R. (1998). *CIF applications. VII. CYCLOPS2: extending the validation of CIF data names*. *J. Appl. Cryst.* **31**, 278–281.

Bluhm, W. (2000). *STAR (CIF) parser*. http://pdb.sdsc.edu/STAR/index.html.

Brown, I. D., Zabobonin, A. & Holt, B. (2004). *beCIF. Browser and editor for CIF.* Private communication.

Collaborative Computational Project, Number 4 (1994). *The CCP4 suite: programs for protein crystallography. Acta Cryst.* D**50**, 760–763.

Edgington, P. R. (1997). *HICCuP: High-Integrity CIF Checking using Python*. Cambridge: Cambridge Crystallographic Data Centre.

Greer, D. S. (2000). *Macromolecular structure RFP response*. Revised submission. http://openmms.sdsc.edu/OpenMMS-1.5.1_Std/openmms/docs/specs/lifesci_00-11-01.pdf.

Greer, D. S., Westbrook, J. D. & Bourne, P. E. (2002). *An ontology driven architecture for derived representations of macromolecular structure. Bioinformatics*, **18**, 1280–1281.

Hall, S. R. (1993). *CIF applications. III. CYCLOPS: for validating CIF data names*. *J. Appl. Cryst.* **26**, 480–481.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Bernstein, H. J. (1996). *CIF applications. V. CIFtbx2: extended tool box for manipulating CIFs*. *J. Appl. Cryst.* **29**, 598–603.

Hall, S. R. & Sievers, R. (1993). *CIF applications. I. QUASAR: for extracting data from a CIF*. *J. Appl. Cryst.* **26**, 469–473.

Hester, J. R. (2002). *PyCifRW. A general CIF reading/writing utility in Python.* http://www.ansto.gov.au/natfac/ANBF/CIF/.

Hester, J. R. & Okamura, F. P. (1998). *CIF applications. X. Automatic construction of CIF input functions: CifSieve*. *J. Appl. Cryst.* **31**, 965–968.

Knuth, D. E. (1986). *The TEXbook*. *Computers and Typesetting*, Vol. A. Reading, MA: Addison-Wesley.

McMahon, B. (1993). *ciftex: translation utility from CIF to TEX.* ftp://ftp.iucr.org/pub/ciftex.tar.Z.

McMahon, B. (1998). *vcif: a utility to validate the syntax of a Crystallographic Information File.* http://www.iucr.org/iucr-top/cif/software/vcif/index.html.

OMG (2001). *Life Sciences Research Domain Task Force.* http://www.omg.org/homepages/lsr/.

Ousterhout, J. K. (1994). *Tcl and the Tk toolkit*. Reading, MA: Addison-Wesley.

Painter, J. & Merritt, E. A. (2004). *mmLib Python toolkit for manipulating annotated structural models of biological macromolecules. J. Appl. Cryst.* **37**, 174–178.

Patel, A. J. (2002). *Yapps: Yet Another Python Parser System.* http://theory.stanford.edu/∼amitp/yapps/.

Rossum, G. van (1991). *Python programming language*. http://www.python.org.

Spadaccini, N. & Hall, S. R. (1994). *Star_Base: accessing STAR File data. J. Chem. Inf. Comput. Sci.* **34**, 509–516.

Stampf, D. R. (1994). *ZINC: galvanizing CIF to work with UNIX.* Brookhaven: Protein Data Bank.

Toby, B. H. (2003). *CIF applications. XIII. CIFEDIT, a program for viewing and editing CIFs*. *J. Appl. Cryst.* **36**, 1288–1289.

Tosic, O. & Westbrook, J. D. (2000). *CIFParse. A library of access tools for mmCIF.* Reference guide. http://pdb.rutgers.edu/mmcif/CIFPARSE-OBJ/cifparse/index.html.

Wall, L., Schwartz, R. L., Christiansen, T. & Orwant, J. (2000). *Programming Perl*, 3rd ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc.

Westbrook, J. D. & Bourne, P. E. (2000). *STAR/mmCIF: an ontology for macromolecular structure*. *Bioinformatics*, **16**, 159–168.

Westbrook, J. & Fitzgerald, P. (2003). *The PDB format, mmCIF formats and other data formats. Structural bioinformatics*, edited by P. E. Bourne & H. Weissig, pp. 161–179. Hoboken, NJ: John Wiley & Sons, Inc.

Westbrook, J. D., Hsieh, S.-H. & Fitzgerald, P. M. D. (1997). *CIF applications. VI. CIFLIB: an application program interface to CIF dictionaries and data files. J. Appl. Cryst.* **30**, 79–83.

Westrip, S. P. (2004). *printCIF for Word.* http://www.iucr.org/iucr-top/cif/software/printCIFforWord/index.html.

Winn, M. (1998). *cif.el: an Emacs mode for CIF*. Daresbury Laboratory, Warrington, England.

5.4. *CIFtbx*: Fortran tools for manipulating CIFs

BY H. J. BERNSTEIN AND S. R. HALL

5.4.1. Introduction

CIFtbx is a function library for programmers developing CIF applications. It is written in Fortran and is intended for use with Fortran programs. The first version was released in 1993 (Hall, 1993*b*) and was extended (Hall & Bernstein, 1996) to accommodate subsequent CIF applications and DDL changes. The *CIFtbx* library is for novice and expert programmers of CIF applications. It has been used to develop CIF manipulation programs such as *CYCLOPS* (Bernstein & Hall, 1998), *CIFIO* (Hall, 1993*a*), *cif2cif* (Bernstein, 1997), *pdb2cif* (Bernstein *et al.*, 1998) and *cif2pdb* (Bernstein & Bernstein, 1996). Programmers writing in C, C++ and mixed Fortran–C should consider alternative approaches, as discussed in Chapter 5.1 or in the work on *CCP4* (Keller, 1996).

The description of library functions below assumes familiarity with the STAR, CIF and DDL syntax described in Part 2. A complete *Primer and reference manual* for *CIFtbx* is provided on the CD-ROM accompanying this volume.

Fortran is a very general and powerful language, and many compilers allow programming in a wide variety of styles. However, there is a traditional Fortran programming style that ensures portability to a wide variety of platforms. *CIFtbx* conforms to this style and has been ported to many platforms. The internals of *CIFtbx* and the style chosen are discussed at the end of this chapter and in more detail in the *Primer*.

5.4.2. An overview of the library

The *CIFtbx* library is made up of functions, subroutines and variables that can be added to application programs as 'commands' to read and write CIF data. They may also be used to automatically validate incoming and outgoing CIF data. The self-checking aspects of some functions ensure that data are syntactically correct and, when used with DDL dictionaries, that individual items conform to their formal definitions.

The *CIFtbx* commands are invoked in user software as standard Fortran function or subroutine calls. For example, to open the dictionary file 'core.dic' one uses the *logical* function `dict_` as follows.

```
FN = dict_('core.dic','valid')
```

The argument `'core.dic'` is the local file identifier for the relevant dictionary. The argument `'valid'` signals that checking should be done against the data definitions in this dictionary. The local *logical* variable `FN` is returned as `.true.` if `dict_` opens the file core.dic correctly; otherwise the function is returned as `.false.`.

Some *CIFtbx* commands are issued as subroutine calls. For example, to clear the internal data tables the programmer inserts the command

```
call purge_
```

The arguments in *CIFtbx* commands have been kept to a minimum. Most of the parameter setting is handled automatically by reading and setting variables held in common blocks supplied as the file ciftbx.cmn. The type declarations for all the commands are also provided in the file ciftbx.cmn, and the programmer must '`include`' this file in each application program, function or subroutine invoking *CIFtbx* commands.

The flexibility of the CIF syntax can present some challenges to an author of applications reading or writing CIF data. This is because the information in a CIF may be in any order, have data names as either upper or lower case, and have an arbitrary spacing between data items. For example, one may extract the cell parameters from the front of a CIF and place them at the end, change all the data names from lower case to upper case, and introduce a blank line between each data name and its value, and yet the data (value) content of the output CIF will be identical to that input. *CIFtbx* provides the application writer with the tools to handle such presentation details seamlessly without altering the basic information content.

Most importantly, *CIFtbx* allows applications to be 'object-oriented', in that data items are simply requested by name without prior knowledge of the file structure. It also allows for more advanced data processing in which data items are parsed sequentially, and typed and validated *via* the dictionary. This enables items to be read independently of the names, and the data typing is automatically determined and returned. In this way, where needed, applications can go beyond the position-independent context of a CIF.

The main purpose of *CIFtbx* is to manipulate CIF data. However, there is much in common between CIF and the Extensible Markup Language XML (Bray *et al.*, 1998), and facilities have been added to *CIFtbx* to facilitate writing output in XML as well as CIF format.

CIFtbx provides four basic kinds of facilities for programmers:
(i) commands to *initialize* later handling;
(ii) commands to *read* CIF data;
(iii) commands to *write* CIF data;
(iv) variables for *monitor* and *control* signals.
These commands are described in detail below.

5.4.3. Initialization commands

Initialization commands are applied at the start of a program to set global conditions for processing CIF data. There are only two commands of this type.

```
logical function init_
  (devcif, devout, devdir, deverr)
  integer devcif, devout, devdir, deverr
logical function dict_ (fname, checks)
  character fname*(*), checks*(*)
```

`init_` is an optional command that specifies the device number assignments for the input CIF *`devcif`*, the output CIF *`devout`*, an internal scratch file *`devdir`* and the file containing error messages *`deverr`*. The internal scratch file *`devdir`* is used to hold a copy of

Affiliations: HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia.

the input CIF as a direct-access file (*i.e.* for random access to parts of the CIF). `init_` is a *logical* function that is always returned with a value of `.true.`. The default device numbers for these files are 1, 2, 3 and 6.

`dict_` is an optional command for opening a dictionary *`fname`* and initiating various optional data checks, *`checks`*. The choices of checks to perform are given by a string of blank-separated five-character 'check codes', such as `valid` or `dtype`, which turn on checking for the validity of tags or types of values, respectively. `dict_` is a *logical* function which is returned as `.true.` if the named dictionary was opened and if the check codes are recognizable.

5.4.4. Read commands

These commands are used to read data from an existing CIF. Since CIF data are order-independent, most applications would work from a known list of data names (tags) and extract the desired values from the CIF in the order specified. However, some applications need to browse a CIF in the order of presentation. In *CIFtbx*, a blank name has the meaning of the next name in the file.

```
logical function ocif_ (fname)
  character fname*(*)
logical function data_ (name)
  character name*(*)
logical function bkmrk_ (mark)
  integer mark
logical function find_ (name, type, strg)
  character name*(*), type*(*), strg*(*)
logical function test_ (name)
  character name*(*)
logical function name_ (name)
  character name*(*)
logical function numb_ (name, numb, sdev)
  character name*(*)
  real numb, sdev
logical function numd_ (name, numb, sdev)
  character name*(*)
  double precision numb, sdev
logical function char_ (name, strg)
  character name*(*), strg*(*)
logical function cmnt_ (strg)
  character strg*(*)
subroutine purge_
```

`ocif_` requests the named CIF *`fname`* to be opened. The *logical* function is returned as `.true.` if the CIF can be opened.

`data_` specifies the data block *`name`* containing the data to be read from the CIF. The *logical* function is returned as `.true.` if the data block is found.

`bkmrk_` is a bookmark function that saves or restores the current position in the CIF so that data can be accessed nonsequentially if need be. The *logical* function is returned as `.true.` if there is space to store the current position or if the restored bookmark number is valid.

`find_` finds the requested item in the current data block. The *logical* function is returned as `.true.` if the item is found.

`test_` provides the data attributes of a data item in the current data block. The *logical* function is returned as `.true.` if the item is found. The data attributes are returned in the common-block variables `list_`, `type_`, `dictype_`, `diccat_` and `dicname_`.

`name_` identifies the next data name in the current data block. The *logical* function is returned as `.true.` if another data name exists in the data block and `.false.` if the end of the data block is reached. The name is returned in the function argument, *`name`*.

`numb_` returns the number *`numb`* and its standard uncertainty *`sdev`* (if appended) of a named data item *`name`*. The *logical* function is returned as `.true.` if the item is present and is a number. If the item is either absent or cannot be recognized as a valid number, the function is returned as `.false.` and the original numeric argument values are not changed.

`numd_` returns the number *`numb`* and its standard uncertainty *`sdev`* (if appended) as double-precision variables of a named data item *`name`*. The *logical* function is returned as `.true.` if the item is present and is a number. If the item is either absent or cannot be recognized as a valid number, the function is returned as `.false.` and the original numeric argument values are not changed.

`char_` returns character or text strings, *`strg`*, of the named data item *`name`*. The *logical* function is returned as `.true.` if the item is present. If text lines are being read, this function is called repeatedly until the *logical variable* `text_` is `.false.`.

`cmnt_` returns the next comment, *`strg`*, in the current data block. The *logical* function is returned as `.true.` if a comment is present. The initial comment character '`#`' is not included in the returned string and a completely blank line is treated as a comment.

`purge_` closes all attached data files and clears all tables and pointers. This is a subroutine call.

5.4.5. Write commands

The following commands are available for writing data to a new CIF.

```
logical function pfile_ (fname)
  character fname*(*)
logical function pdata_ (name)
  character name*(*)
logical function ploop_ (name)
  character name*(*)
logical function pnumb_ (name, numb, sdev)
  character name*(*)
  real numb, sdev
logical function pnumd_ (name, numb, sdev)
  character name*(*)
  double precision numb, sdev
logical function pchar_ (name, string)
  character name*(*), string*(*)
logical function pcmnt_ (string)
  character string*(*)
logical function ptext_ (name, string)
  character name*(*), string*(*)
logical function prefx_ (strg, lstrg)
  character strg*(*)
  integer lstrg
subroutine close_
```

`pfile_` creates a new file with the specified file name *`fname`*. The *logical* function is returned as `.true.` if the file is opened. The value will be `.false.` if the file already exists.

`pdata_` puts the string **`data_`*`name`*** from the argument *`name`* into the output CIF. The *logical* function is returned as `.true.` if the block is created. The value will be `.false.` if the block name already exists. This command inserts the string **`save_`*`name`*** instead of the data-block name if the variable `saveo_` is set to `.true.`. If the prior block was a save frame, the necessary terminal **`save_`** is written for that block before the new block is started.

`ploop_` puts the specified data name *`name`* into the output CIF. On the first invocation of this command for a given loop, a **`loop_`** string is placed before the data name. The *logical* function is returned as `.true.` if the name passes any requested dictionary validation checks. Once a series of data names for a **`loop_`** header has been declared by calls to this function, all calls to `pchar_`, `ptext_`, `pnumb_` or `pnumd_` for the associated data values must be made with *blank* data names or the **`loop_`** will be terminated. (At the very least, the first character of these data names must be blank.)

`pchar_` puts the specified data name *`name`* and *character string* *`string`* into the output CIF. If the data name is blank, only the

character string is put. The *logical* function is returned as `.true.` if the data name passes any requested dictionary validation checks.

`pnumb_` puts the specified data name *`name`*, single-precision number *`numb`* and an appended standard uncertainty *`sdev`* into the output CIF. The *logical* function is returned as `.true.` if the data name passes any requested dictionary validation checks.

`pnumd_` puts the specified data name *`name`*, double-precision number *`numb`* and an appended standard uncertainty *`sdev`* into the output CIF. The *logical* function is returned as `.true.` if the data name passes any requested dictionary validation checks.

`ptext_` puts the specified data name *`name`* and text string *`string`* into the output CIF. The data name will only be inserted on the first invocation of a sequence. The *logical* function is returned as `.true.` if the data name passes any requested dictionary validation checks. This command must be invoked repeatedly until the text is finished. The terminal semicolon character '`;`' is placed in the output CIF when the next call to `pchar_`, `pnumb_` or `pnumd_` is made, or if a call is made to `ptext_` for a different data name.

`pcmnt_` puts the specified comment string *`string`* into the output CIF. The *logical* function is always returned as `.true.`. The comment character '`#`' should not be included in the string. A blank comment is presented as a blank line without the leading '`#`'. The string `char(0)//char(0)` can be used to produce an empty comment with the leading '`#`'.

`prefx_` prefixes the specified string *`strg`* of length *`lstrg`* to subsequent lines of the output CIF. The total line length is still limited to the value given by the variable `line_` (the default is 80 characters). This function is useful when embedding a CIF into another text document, such as a PDB REMARK. The *logical* function is always returned as `.true.`.

`close_` closes the output CIF only. This command *must* be used if `pfile_` is used. This a subroutine call.

5.4.6. Variables

The *CIFtbx* library also contains a large number of variables declared in the common blocks in the file ciftbx.cmn that provide signals to the programmer on various aspects of the data reading and writing processes. These variables are described below in four broad categories, as shown in Table 5.4.6.1: general monitor variables, general control variables, input monitor variables and output control variables.

Note that for all but special applications only the basic variables `list_`, `loop_`, `strg_`, `text_` and `type_` are usually used. These variables supplement the argument lists of the various commands, providing essential status information.

5.4.6.1. General monitor variables

These variables are returned by *CIFtbx* and provide information about the general status of processing.

`file_`: *character string* containing the file name of the current input file.

`longf_`: *integer variable* containing the length of the file name in *`file`*.

`precn_`: *integer variable* containing the line number (starting from 1) of the last line written to the output CIF.

`recn_`: *integer variable* containing the line number (starting from 1) of the last line read from the input CIF.

`tbxver_`: *character*32 variable* that is the *CIFtbx* version and date in the form *`'CIFtbx version N.N.N DD MMM YYYY'`* (some older versions of *CIFtbx* use a two-digit year and have a comma after the version number).

Table 5.4.6.1. *CIFtbx variables*

General monitor	General control	Input monitor	Output control
	`alias_`		`aliaso_`
			`align_`
	`append_`		
		`bloc_`	
		`decp_`	`pdecp_`
		`diccat_`	
		`dicname_`	
		`dictype_`	
		`dicver_`	
			`esdlim_`
`file_`			
		`glob_`	`globo_`
	`line_`		
		`list_`	
		`long_`	
`longf_`			
		`loop_`	
		`lzero_`	`plzero_`
	`nblank_`		`nblanko_`
		`posdec_`	`pposdec_`
		`posend_`	`pposend_`
		`posnam_`	`pposnam_`
		`posval_`	`pposval_`
`precn_`			
		`quote_`	`pquote_`
	`recbeg_`		
	`recend_`		
`recn_`			
		`save_`	`saveo_`
		`strg_`	
			`tabl_`
	`tabx_`		`ptabx_`
		`tagname_`	
`tbxver_`			
		`text_`	
		`type_`	
			`xmlout_`
			`xmlong_`

5.4.6.2. General control variables

These variables control *CIFtbx* commands. The user may accept the default values or may store new values into these variables to change the behaviour of the commands.

`alias_`: *logical variable* to control the use of data-name aliases for input items. If set to `.true.`, aliases from the input dictionary may be used (see Section 5.4.7). The default is `.true.`.

`append_`: *logical variable* to control reuse of the direct-access file. If set to `.true.`, it will cause each call to `ocif_` to append the information found to the current CIF. The default is `.false.`.

`line_`: *integer variable* to set the input/output line limit for processing a CIF. The default value is 80 characters. This limit counts the visible printable characters of the line, not the system-dependent line terminators.

`nblank_`: *logical variable* to control the treatment of input blank strings. If set to `.true.`, `char_` or `test_`, it will return the type as 'null' rather than 'char' when encountering a quoted blank.

`recbeg_`: *integer variable* to give the record number of the first record to be used. May be changed by the user to restrict access to a CIF.

`recend_`: *integer variable* to give the record number of the last record to be used. May be changed by the user to restrict access to a CIF.

`tabx_`: *logical variable* is set to `.true.` for tab stops to be expanded to blanks during the reading of a CIF. The default is `.true.`.

5.4.6.3. Input monitor variables

These variables are returned by *CIFtbx* tools and are used to decide on subsequent actions in the program. The lengths of the character strings that hold data names and block names are controlled by the parameter NUMCHAR in the common-block declarations.

bloc_: *character string* containing the current data-block name.

decp_: *logical variable* is .true. if a decimal point is present in the input numeric value.

diccat_: *character string* containing the category name specified in the attached dictionaries.

dicname_: *character string* containing the root alias data name (see Section 5.4.7) specified in the attached dictionaries or, after a call to dict_, the name of the dictionary.

dictype_: *character string* containing the data-type code specified in the attached dictionaries. These types may be more specific (*e.g.* 'float' or 'int') than the types given by the variable type_ (*e.g.* 'numb').

dicver_: *character string* containing the version of a dictionary after a call to dict_.

glob_: *logical variable* is .true. if the current data block is a global block. The application is responsible for managing the relationship of global data to other data blocks.

list_: *integer variable* containing the sequence number of the current looped list. This value may be used by the application to identify variables that are in different lists or that are not in a list (a zero value).

long_: *integer variable* containing the length of the data string in strg_.

loop_: *logical variable* is .true. if another loop packet is present in the current looped list.

lzero_: *logical variable* is .true. if the input numeric value is of the form [*sign*]0.*nnnn* rather than [*sign*].*nnnn*.

posdec_: *integer variable* containing the column number (position along the line, counting from 1 at the left) of the decimal point for the last number read.

posend_: *integer variable* containing the column number (position along the line, counting from 1 at the left) of the last character for the last string or number read.

posnam_: *integer variable* containing the starting column (position along the line, counting from 1 at the left) of the last name or comment read.

posval_: *integer variable* containing the starting column (position along the line, counting from 1 at the left) of the last data value read.

quote_: *character variable* giving the quotation symbol found delimiting the last string read.

save_: *logical variable* is .true. if the current data block is a save frame, otherwise .false..

strg_: *character variable* containing the data name or string representing the data value last retrieved.

tagname_: *character variable* containing the data name of the current data item as it was found in the CIF. May differ from dicname_ because of aliasing.

text_: *logical variable* is .true. if another text line is present in the current input text block.

type_: *character variable* containing the data-type code of the current input data item. This will be one of the four-character strings 'null' (for missing data, the period or the question mark), 'numb' (for numeric data), 'char' (for most character data) or 'text' (for semicolon-delimited multi-line character data). For most purposes the type 'text' is a subtype of the type 'char', not a distinct data type. *CIFtbx* permits multi-line text fields to be used whenever character strings are expected.

5.4.6.4. Output control variables

These variables are specified to control the processing by *CIFtbx* commands that write CIFs.

aliaso_: *logical variable* to control the use of data-name aliases for output items. If set to .true., preferred synonyms from the input dictionary may be output (see Section 5.4.7). The default is .false..

align_: *logical variable* to control the column alignment of data values in **loop_** lists output to a CIF. The default is .true..

esdlim_: *integer variable* to set the upper limit of appended standard uncertainty (e.s.d.) integers output by **pnumb_**. The default value is 19, which limits standard uncertainties to the range 2–19.

globo_: *logical variable* which if set to .true. will cause the output data block from pdata_ to be written as a global block.

nblanko_: *logical variable* controls the treatment of output blank strings. If set to .true., output quoted blank strings will be converted to an unquoted period (*i.e.* to a data item of type null). Recall that *CIFtbx* treats an unquoted period or question mark as being of type null.

pdecp_: *logical variable* controls the treatment of output decimal numbers. If set to .true., a decimal point will be inserted into numbers output by pnumb_ or pnumbd_. If set to .false., a decimal point will be output only when needed. The default is .false..

plzero_: *logical variable* controls the treatment of leading zeros in output decimal numbers. If set to .true., a zero will be inserted before a leading decimal point. The default is .false..

pposdec_: *integer variable* to set the column number (position along the line, counting from 1 at the left) of the decimal point for the next number to be output.

pposend_: *integer variable* to set the position of the ending column for the next number or *character string* to be output. Used to pad with zeros or blanks.

pposnam_: *integer variable* to set the starting column of the next name or comment to be output.

pposval_: *integer variable* to set the position of the starting column of the next data value to be output.

pquote_: *character variable* containing the quotation symbol to be used for the next string written.

saveo_: *logical variable* is set to .true. for pdata_ to output a save frame, otherwise a data block is output.

ptabx_: *logical variable* is set to .true. for tab stops to be expanded to blanks during the creation of a CIF. The default is .true..

tabl_: *logical variable* is set to .true. for tab stops to be used in the alignment of output data. The default is .true..

xmlout_: *logical variable* is set to .true. to change the output style to XML conventions. Note that this is not a CML (Murray-Rust & Rzepa, 1999) output, but a literal translation from the input CIF. The default is .false..

xmlong_: *logical variable* is set to .true. to change the style of XML output if xmlout_ is .true.. When .true. (the default), XML tag names are the full CIF tag names with the leading underscore, _, removed. When .false., an attempt is made to strip the leading category name as well.

5.4.7. Name aliases

CIF dictionaries written in DDL2 permit data names to be aliased or equivalenced to other data names. This serves two purposes. First, it allows for the different data-name structures used in DDL1

```
    read(8,'(a)',end=400) name
       f1 = test_(name)
       write(6,'(2(3x,a32)') name,dicname_
       name=dicname_
       f1 = test_(name)
       write(6,'(2(3x,a32)') name,tagname_
```

Fig. 5.4.7.1. Example application accessing aliased data names.

and DDL2 dictionaries, and, second, it links equivalent data names within the DDL2 dictionary. Aliasing also allows the use of synonyms appropriate to the application.

CIFtbx is capable of handling aliased data names transparently so that both the input CIF and the application software can use any of the equivalent aliased names. In addition, an output CIF may be written with the data names specified in the *CIFtbx* functions or with names that have been automatically converted to preferred dictionary names. If more than one dictionary is loaded, the first aliases have priority. We call the preferred dictionary name the 'root alias'.

The default behaviour of *CIFtbx* is to accept all combinations of aliases and to produce output CIFs with the exact names specified in the user calls. The interpretation of aliased data names is modified by setting the *logical variable*s `alias_` and `aliaso_`. When `alias_` is set to `.false.`, the automatic recognition and translation of aliases stops. When `aliaso_` is set to `.true.`, the automatic conversion of user-supplied names to dictionary-preferred alias names in writing data to output CIFs is enabled. The preferred alias name is stored in the variable `dicname_` following any invocation of a getting function, such as `numb_` or `test_`. If `alias_` is set to `.false.`, `dicname_` will correspond to the called name. The variable `tagname_` is always set to the actual name used in an input CIF.

For example, the data name **`_atom_site_anisotrop.u[1][1]`** in the DDL2 mmCIF dictionary is aliased to the data name **`_atom_site_aniso_U_11`** in the DDL1 core CIF dictionary. In the example application of Fig. 5.4.7.1, showing the *CIFtbx* function `test_` run with both the mmCIF and core dictionaries loaded, the specified data name **`_atom_site_aniso_U_11`** is used to inquire as to the names used in an input CIF.

The execution of this code results in the following printout.

```
_atom_site_aniso_U_11
          _atom_site_anisotrop.u[1][1]
_atom_site_anisotrop.u[1][1]
          _atom_site_aniso_U_11
```

5.4.8. Implementation of the tools

Implementation of the *CIFtbx* tools is straightforward. The supplied source files in all versions are: ciftbx.f, ciftbx.sys (used in ciftbx.f) and ciftbx.cmn (used in local applications). More recent versions of *CIFtbx* (version 2.4 and later) require certain additional source files: ciftbx.cmf, ciftbx.cmv, hash_funcs.f and clearfp.f (or clearfp_sun.f).

The common file ciftbx.cmn must be '`include`'d into any local routines that use *CIFtbx* tools. The library in ciftbx.f may be invoked by either (i) compiling and linking the resulting object file as an object library, or with explicit references in the application linking sequence (versions 2.4 and later require hash_funcs.o as an additional object file); or (ii) including ciftbx.f in the local application and compiling and linking it with the local program.

Approach (i) is more efficient, but for some applications approach (ii) may be simpler.

5.4.9. How to read CIF data

The *CIFtbx* approach to reading CIF data is illustrated using a simple example program *CIF_IN* (Fig. 5.4.9.1), which reads the file test.cif (Fig. 5.4.9.2) and tests the input data items against the dictionary file cif_core.dic. The resulting output is shown in Fig. 5.4.9.3.

The program *CIF_IN* may be divided into the following steps, each tagged with the relevant reference letter in the comment records of the listing shown in Fig. 5.4.9.1.

A: Define the local variables. The *CIFtbx* variables are added with the line `include 'ciftbx.cmn'`.

B: Assign device numbers to the files using the command `init_`. The device number 1 refers to the input CIF, 3 to the scratch file and 6 (stdout) to the error-message files. The device number 2 refers to an output CIF, if we were to choose to write one.

C: Open a specific dictionary file named cif_core.dic with the command `dict_`. The code `valid` signals that the input data items are to be validated against the dictionary. In this application, `dict_` is invoked in an `IF` statement that tests whether the command is successful.

D: Open the CIF test.cif with the command `ocif_` and test that the file is opened.

E: Invoke the `data_` command, containing a blank block code, to 'open' the next data block. The block name encountered is placed in the variable `bloc_`, which in this application is printed.

F: Read the cell-length values and their standard uncertainties with the `numb_` command, and print these out. Test whether all of the requested data items are found.

G: The `char_` function is used to read a single *character string*.

H: The `name_` function is used to get the data name of the next data item encountered.

I: This sequence illustrates how text lines are read. The `char_` function is used to read each line and the `text_` variable is tested to see whether another text line exists in this data item.

J: This sequence illustrates how a looped list of items is read. Individual items are read using `char_` or `numb_` functions and the existence of another packet of items is tested with the variable `loop_`.

The resulting printout is shown in Fig. 5.4.9.3. In this figure, note the following:

(i) The first six lines of the printout are output by *CIFtbx* routines, not by the program *CIF_IN*. They occur when the `data_` command is executed and data items in the block **`mumbo_jumbo`** are read from the CIF and checked against the dictionary file. Note that this is when the *CIFtbx* routines store the pointers and attributes of all items in the data block. All subsequent commands use these pointers to access the data.

(ii) The '**`####`**' string in front of **`_cell_length_a`** in the input CIF makes this line a comment and makes it inaccessible to *CIF_IN*.

(iii) Data items may be read from a CIF in any order but looped items must normally be in the same list. If one needs to access looped items in different lists simultaneously, the `bkmrk_` command is used to preserve *CIFtbx* loop pointers.

5.4.10. How to write a CIF

Writing a CIF usually is simpler than reading an existing one. An example of a CIF-writing program is shown in Fig. 5.4.10.1. This example is intentionally trivial. The created CIF test.new is shown in Fig. 5.4.10.2. Note that command `dict_` causes all output items to be checked against the dictionary cif_core.dic. Unknown names are flagged in the output CIF with the comment

```
                        PROGRAM  CIF_IN
C
C....A.. Define the data variables
         include       'ciftbx.cmn'
         logical       f1,f2,f3
         character*32  name
         character*80  line
         real          cela,celb,celc,siga,sigb,sigc
         real          x,y,z,u,numb,sdev
         data          cela,celb,celc,siga,sigb,sigc /6*0.0/

C....B.. Assign the CIFtbx files
         f1 = init_( 1, 2, 3, 6 )

C....C.. Request dictionary validation check
         if(dict_('cif_core.dic','valid'))  goto 100
         write(6,'(/a/)') ' Requested Core dictionary not present'

C....D.. Open the CIF to be accessed
100      name='test.cif'
         if(ocif_(name))      goto 120
         write(6,'(a///)')  ' >>>>>>>>> CIF cannot be opened'
         stop
C....E.. Assign the data block to be accessed
120      if(.not.data_(' '))       goto 200
         write(6,'(/a,a/)') ' Access items in data block  ',bloc_

C....F.. Extract some cell dimensions; test all is OK
         f1 = numb_('_cell_length_a', cela, siga)
         f2 = numb_('_cell_length_b', celb, sigb)
         f3 = numb_('_cell_length_c', celc, sigc)
         if(.not.(f1.and.f2.and.f3)) write(6,'(a)') ' Cell lengths missing!'
         write(6,'(a,6f10.4)') ' Cell ',cela,celb,celc,siga,sigb,sigc

C....G.. Extract space group notation (expected char string)
         f1 = char_('_symmetry_cell_setting', name)
         write(6,'(a,a/)') ' Cell setting   ',name(1:long_)

C....H.. Get the next name in the CIF and print it out
         f1 = name_(name)
         write(6,'(a,a/)') ' Next data name in CIF is   ',name

C....I.. List the audit record (possible text line sequence)
         write(6,'(a)') ' Audit record'
140      f1 = char_('_audit_update_record', line)
         write(6,'(a)') line
         if(text_)  goto 140

C....J.. Extract atom site data in a loop
         write(6,'(/a)') ' Atom sites'
160      f1 = char_('_atom_site_label', name)
         f2 = numb_('_atom_site_fract_x', x, sx)
         f2 = numb_('_atom_site_fract_y', y, sy)
         f2 = numb_('_atom_site_fract_z', z, sz)
         f3 = numb_('_atom_site_U_iso_or_equiv', u, su)
         write(6,'(1x,a4,4f8.4)') name,x,y,z,u
         if(loop_)  goto 160
C
         goto 120
200      continue
         end
```

Fig. 5.4.9.1. Sample program *CIF_IN*. See text for explanation.

`'#< not in dictionary'`. This applies to both looped and single data items.

A more complex example of writing a CIF is given in the program *cif2cif* available with the *CIFtbx* release. A similar program that reads a CIF and writes an XML file is *cif2xml*, also available with the *CIFtbx* release.

5.4.11. Error-message glossary

The *CIFtbx* routines will generate explicit error messages in the printout or in the created CIF if requested to do so (*e.g.* during dictionary checks). If data processing cannot continue (*i.e.* fatal errors), an appropriate error message is placed in the printout and execution terminates. However, the default approach is to remain mute and for error detection to be monitored by the application program *via* the *CIFtbx* functions returning `.true.` or `.false.` values that tell the application program whether the command was performed correctly. This places the primary responsibility for error checking on the application software. The importance of this approach is that it enables the local application to respond to run-time problems in a controlled way and to take corrective action if it is possible. However, some types of processing errors, such as exceeding the dimensions of critical *CIFtbx* arrays, do require appropriate messages to be issued and for execution to cease.

```
data_mumbo_jumbo

_audit_creation_date               91-03-20
_audit_creation_method             from_xtal_archive_file_using_CIFIO
_audit_update_record
; 91-04-09           text and data added by Tony Willis.
  91-04-15           rec'd by co-editor with diagram as manuscript HL7
;
_dummy_test     "rubbish to see what dict_ says"

_chemical_name_systematic
    trans-3-Benzoyl-2-(tert-butyl)-4-(iso-butyl)-1,3-oxazolidin-5-one
_chemical_formula_moiety            'C18 H25 N O3'
_chemical_formula_weight            303.40
_chemical_melting_point             ?

####_cell_length_a                      5.959(1)
_cell_length_b                      14.956(1)
_cell_length_c                      19.737(3)
_cell_measurement_theta_min         25
_cell_measurement_theta_max         31
_symmetry_cell_setting              orthorhombic

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
  s  .20200  .79800  .91667  .030(3)  Uij  ?
  o  .49800  .49800  .66667  .02520  Uiso  ?
  c1  .48800  .09600  .03800  .03170  Uiso  ?

loop_ _blat1 _blat2 1 2 3 4 5 6 a b c d 7 8 9 0
```

Fig. 5.4.9.2. Example CIF read by the sample program *CIF_IN* shown in Fig. 5.4.9.1.

```
 CIFtbx warning: test.cif data_mumbo_jumbo line:     8
 Data name _dummy_test not in dictionary!
 CIFtbx warning: test.cif data_mumbo_jumbo line:     35
 Data name _blat1 not in dictionary!
 CIFtbx warning: test.cif data_mumbo_jumbo line:     35
 Data name _blat2 not in dictionary!

 Access items in data block  mumbo_jumbo

 Cell dimension(s) missing!
 Cell     0.0000   14.9560   19.7370   0.0000    0.0010    0.0030
 Cell setting   orthorhombic

 Next data name in CIF is   _atom_type_symbol

 Audit record
  91-04-09           text and data added by Tony Willis.
  91-04-15           rec'd by co-editor with diagram as manuscript HL7

 Atom sites
 s     0.2020  0.7980  0.9167  0.0300
 o     0.4980  0.4980  0.6667  0.0252
 c1    0.4880  0.0960  0.0380  0.0317
```

Fig. 5.4.9.3. Printout from the example program *CIF_IN* run on the test file of Fig. 5.4.9.2.

CIFtbx error messages are in four parts: 'warning' or 'error' header line, the name of the file being processed, the current data block or save frame, and the line number. Another line contains the text of the message.

5.4.11.1. Fatal errors: array bounds

The following fatal messages are issued if the *CIFtbx* array bounds are exceeded. Operation terminates immediately. Array bounds can be adjusted by changing the PARAMETER values in ciftbx.sys. If the value of MAXBUF needs to be changed, the file ciftbx.cmv must also be updated.

```
Input line_ value > MAXBUF
Number of categories > NUMBLOCK
Number of data names > NUMBLOCK
Cifdic names > NUMDICT
Dictionary category names > NUMDICT
Items per loop packet > NUMITEM
Number of loop_s > NUMLOOP
```

```
C....... Open a new CIF
400      if(pfile_('test.new'))  goto 450
         write(6,'(//a/)') ' Output CIF by this name exists already!'
         goto 500
C....... Request dictionary validation check
450      if(dict_('cif_core.dic','valid'))  goto 460
         write(6,'(/a/)') ' Requested Core dictionary not present'
C....... Insert a data block code
460      f1 = pdata_('whoops_a_daisy')
C....... Enter various single data items to show how
         f1 = pchar_('_audit_creation_method','using CIFtbx')
         f1 = pchar_('_audit_creation_extra2',"Terry O'Connell")
         f1 = pchar_('_audit_creation_extra3','Terry O"Connell')
         f1 = ptext_('_audit_creation_record',' Text data may be ')
         f1 = ptext_('_audit_creation_record',' entered like this')
         f1 = ptext_('_audit_creation_record',' or in a loop.')
         f1 = pnumb_('_cell_measurement_temperature', 293., 0.)
         f1 = pnumb_('_cell_volume', 1759.0, 13.)
         f1 = pnumb_('_cell_length_b', 8.75353553524313,0.)
         f1 = pnumb_('_cell_length_c', 19.737, .003)
C....... Enter some looped data
         f1 = ploop_('_atom_type_symbol')
         f1 = ploop_('_atom_type_oxidation_number')
         f1 = ploop_('_atom_type_number_in_cell')
         do 470 i=1,3
         f1 = pchar_(' ',alpha(1:i))
         f1 = pnumb_(' ',float(i),float(i)*0.1)
470      f1 = pnumb_(' ',float(i)*8.64523,0.)
C....... Do it again but as contiguous data with text data
         f1 = ploop_('_atom_site_label')
         f1 = ploop_('_atom_site_occupancy')
         f1 = ploop_('_some_silly_text')
         do 480 i=1,2
         f1 = pchar_(' ',alpha(1:i))
         f1 = pnumb_(' ',float(i),float(i)*0.1)
480      f1 = ptext_(' ',' Hi Ho the diddly oh!')
500      call close_
```

Fig. 5.4.10.1. Sample program to create a CIF.

```
data_whoops_a_daisy
_audit_creation_method      'using CIFtbx'
_audit_creation_extra2      'Terry O'Connell'    #< not in dictionary
_audit_creation_extra3      'Terry O"Connell'    #< not in dictionary
_audit_creation_record
;Text data may be
 entered like this
 or in a loop.
;
_cell_measurement_temperature   293
_cell_volume                    1759(13)
_cell_length_b                  8.75354
_cell_length_c                  19.737(3)
loop_
    _atom_type_symbol
    _atom_type_oxidation_number
    _atom_type_number_in_cell
        a               1.00(10)        8.64523
        ab              2.0(2)          17.2905
        abc             3.0(3)          25.9357
loop_
    _atom_site_label
    _atom_site_occupancy
    _some_silly_text                    #< not in dictionary
        a               1.00(10)
;Hi Ho the diddly oh!
;
        ab              2.0(2)
;Hi Ho the diddly oh!
;
```

Fig. 5.4.10.2. Sample CIF created by the example program of Fig. 5.4.10.1.

However, the message

`More than MAXBOOK bookmarks requested`

is not 'fatal', in the sense that the function `bkmrk_` returns `.false.` to permit appropriate action before termination. This is effectively a fatal error for which recompilation with a larger value of `MAXBOOK` is necessary. However, this is usually the result of a logic error in the application, and the error has been made non-fatal to allow the programmer to insert debugging code, if desired. The application should clean up and exit promptly.

5.4.11.2. Fatal errors: data sequence, syntax and file construction

`Dict_ must precede ocif_`
Dictionary files must be loaded before an input CIF is opened because some checking occurs during the CIF loading process.

`Illegal tag/value construction`
Data name (*i.e.* a 'tag') and data values are not matched (outside a looped list). This usually means that a data name immediately follows another data name, or a data value was found without a preceding data name. The most likely cause of this error is the failure to provide '**.**' or '**?**' for missing or unknown data values or a failure to declare a **loop_** when one was intended.

`Item miscount in loop`
Within a looped list the total number of data values must be an exact multiple of the number of data names in the **loop_** header.

`Prior save-frame not terminated`
Save-frame terminator found out of context. Save frames must start with **save_*framecode*** and end with **save_**. These messages will be issued if this does not occur.

`Syntax construction error`
Within a data block or save frame the number of data values does not match the number of data names (ignoring loop structures). This message should occur only if there is an internal logic error in the library. Normally the program will terminate on `Item miscount in loop` first.

`Unexpected end of data`
When processing multi-line text the end of the CIF is encountered before the terminal semicolon.

5.4.11.3. Fatal errors: invalid arguments

The following messages are generated by calls with invalid arguments.

`Call to find_ with invalid arguments`
`Internal error in putnum`

5.4.11.4. Warnings: input errors

`Category <cat-code> first implicitly defined in cif`
The category code in the DDL2 data name is not matched by an explicit definition in the dictionary. This may be intentional but usually indicates a typographical error in the CIF or the dictionary.

`Data name <name> not in dictionary!`
The data item name `<name>` was used in the CIF but could not be found in the dictionary.

`Data block header missing`
No **data_** or **global_** was found when expected.

`Duplicate data item <name>`
Two or more identical data names `<name>` have been detected in a data block or save frame.

`Exponent overflow in numeric input`
`Exponent underflow in numeric input`
The numeric value being processed has an exponent that cannot be processed on this machine. If the string involved is not intended as a number, then surrounding it with quotes may resolve the problem.

`Heterogeneous categories in loop <new cat-code>`
`  vs <old cat-code>`
Looped lists should not contain data items belonging to different categories. This error occurs if the category of a new data item fails to match the category of a prior data item. A special category (none) is used to denote item names for which no category has been declared. Warnings are not issued on this level for a loop for which all data items have no declared category.

`Input line length exceeds line_`
Non-blank characters were found beyond the value given by the variable `line_`. The default value for `line_` is 80 (optionally increased to 2048 in *CIFtbx* 2.7 and later for CIF 1.1 compliance). The extra characters in column positions `line_` + 1 through `MAXBUF` will be processed but the input file may need to be reformatted for use with other CIF-handling programs.

`Missing loop_ items set as DUMMY`
A looped list of output items was truncated with an incomplete loop packet (*i.e.* the number of items did not match the number of **loop_** data names). The missing values were set to the *character string* 'DUMMY'.

`Numb type violated <name>`
The data item `<name>` has been processed with an explicit dictionary type numb, but with a non-numeric value. Note that the values '**?**' or '**.**' will *not* generate this message.

`Quoted string not closed`
Character values may be enclosed by bounding quotes. The strict definition of a 'quoted-string' value is that it must start with a `<wq>` digraph and end with a `<qw>` digraph, where `w` is a white-space character blank or tab and `q` is a single or double quote, and the same type of quote mark is used in the terminal digraph as was used in the initial digraph. This message is issued if these conditions are not met.

5.4.11.5. Warnings: output errors

`Converted pchar_ output to text for <string>`
An attempt was made to write a string with `pchar_` instead of `ptext_`, but the string contains a combination of characters for which `ptext_` must be used.

`ESD less than precision of machine`
`Overflow of esd`
`Underflow of esd`
A call to `pnumb_` or `numb_` was made with values of the number and standard uncertainty (e.s.d) which cannot be presented properly on this machine. A bounding value of 0 or 99999 is used for the e.s.d.

`Invalid value of esdlim_ reset to 19`
In processing numeric output, a value of `esdlim_` less than 9 or greater than 99999 was found. `esdlim_` is then set to 19.

`Missing loop_ name set as _DUMMY`
`Missing loop_ items set as DUMMY`
In processing a **`loop_`**, a dummy string has been inserted for a missing header or value.

`Output CIF line longer than line_`
In outputting a line, the data exceed the limit specified in `line_`. This occurs only if a single data name or a value exceeds this limit.

`Out-of-sequence call to end text block`
The termination of a text block has been invoked before a text block has been started. This can only occur with irregular use of the *CIFtbx* routines rather than the standard interface routines.

`Output prefix may force line overflow`
A prefix string placed in `prefx_` exceeds `line_` less the allowed length of tags.

`Prefix string truncated`
A prefix string specified to `prefx_` is longer than the maximum line length allowed. The prefix string is truncated and processing continues.

5.4.11.6. Warnings: dictionary checks

`Aliases and names in different loops; only using first alias`
If a DDL2 dictionary contains a loop of alias declarations, the corresponding data-name declarations are expected to be in the same loop. Only the first alias name is used.

`Attempt to redefine category for item`
`Attempt to redefine type for item`
If a DDL2 dictionary contains a category or type for a data item that conflicts with an earlier declaration, the first is used.

`Categories and names in different loops`
`Types and names in different loops`
If a DDL2 dictionary contains a loop of category or type declarations, the corresponding data-name declarations are expected to be in the same loop. Only the first category name or type is used.

`Category id does not match block name`
In a DDL2 dictionary, the save-frame code is expected to start with the category name. If a category name within the frame is not within a loop, it is checked against that in the frame code and a warning is issued if these do not match.

`Conflicting definition of alias`
A DDL2 dictionary contains a new declaration of a data-name alias which is in conflict with a previous alias definition. The first alias declaration is used.

`Duplicate definition of same alias`
A DDL2 dictionary contains a new declaration of an alias for a data name which duplicates a previously defined alias pair.

`Item name <name> does not match category name`
If category checking is enabled and the category assigned to an item name does not match the intial characters of the item name, this message is issued. This may indicate a typographical error or a deprecated item in the dictionary.

`Item type <type-code> not recognised`
The DDL2 dictionary type codes are translated to the DDL type codes 'numb', 'char' and 'text'. If an unrecognized type code is found no translation occurs.

`Multiple DDL category definitions`
`Multiple DDL name definitions`
`Multiple DDL type definitions`
`Multiple DDL related item definitions`
`Multiple DDL related item functions`
DDL1 and DDL2 declarations for categories, data names, data types and related items are used in the same data block or save frame.

`Multiple categories for one name`
`Multiple types for one name`
A dictionary contains a loop of category or type definitions and an unlooped declaration of a single data name. The first category or type definition is used.

`No category defined in block <name> and name <name> does not match`
A DDL2 dictionary contains no category for the defined data item and it was not possible to derive an implicit category from the block name. This usually indicates a typographical error in the dictionary.

`No category specified for name <name>`
A dictionary contains categories and category checking is enabled but no category is defined for the named data item.

`No name defined in block`
`No name in the block matches the block name`
These messages are issued if a dictionary save frame or data block contains no name definition or if all the names defined fail to match the block name.

`No type specified for name <name>`
A type code is missing from a dictionary and type checking was requested in the `dict_` invocation.

`One alias, looped names, linking to first`
A DDL2 dictionary may contain a list of data names and a single alias outside this loop. In this case, the correct name to which to link the alias must be derived implicitly. If the save-frame code matches the first name in the loop no warning is issued, because the use of the block name was probably the intended result, but if no such match is found this warning is issued.

5.4.12. Internals and programming style

CIFtbx is programmed in a highly portable Fortran programming style. However, on some older systems, some adaptation may be necessary to allow compilation. Implementors should be aware of the extensive use of variables in common blocks to transmit information and control execution (programming by side-effects), the use of the INCLUDE statement, the use of the ENDDO statement, the names of routines used internally by the package, the use of names longer than six characters and the use of names including the underscore character.

Some aspects of the internal organization of the library to deal with characteristics of CIFs are worth noting. *CIFtbx* copies an input CIF to a direct-access (*i.e.* random-access) file, but writes an output CIF directly. All data names are converted to lower case to deal with the case-insensitive nature of CIF. A hierarchy of parsing routines is used to deal with processing white space.

The major issues of programming style and internals are summarized here. See the *Primer* on the CD-ROM for more information.

5.4.12.1. Programming style

A traditional Fortran style of programming is used in *CIFtbx*. Common blocks are declared to report and control the state of the processing. This allows argument lists to be kept short and avoids the need to create complex data structure types, but introduces extensive 'programming by side-effects'. In order to reduce the impact of this approach on users, two different views of the common blocks are provided. The declarations in ciftbx.cmn are needed by all users. The more extensive declarations in ciftbx.sys, which include the same common declarations as are found in ciftbx.cmn and additional declarations used internally within *CIFtbx*, are provided for use in maintaining the library. Caution is needed in making internal modifications to the library to maintain the desired relationships among the actions of various routines and the states of variables declared in the common blocks.

Statements are written in the first 72 columns of a line, reserving columns one through five for statement labels and using column six for continuation. Approaches that would require the use of C libraries or non-portable Fortran extensions are avoided. For this reason, all the internal service routines are written in Fortran, all memory needed is preallocated with DIMENSION statements and a direct-access file is used to hold the working copy of a CIF.

5.4.12.2. Memory management

Since *CIFtbx* does static memory allocation with DIMENSION statements, it is sometimes necessary to adjust the array dimensions chosen to suit a particular application. It may also be necessary to increase the storage allocated for individual tags to allow for unusually long ones.

The sizes of most arrays and strings used in *CIFtbx* that might require adjustment are controlled by PARAMETER statements in the files ciftbx.sys and ciftbx.cmv (the variable-declaration portion of ciftbx.cmn). The parameters are shown in Table 5.4.12.1.

These values can result in *CIFtbx* requiring more than a megabyte of memory. On smaller machines working with a small dictionary and simple CIFs, considerable space can be saved by reducing the values of NUMDICT and NUMBLOCK.

On the other hand, an application working with several layered dictionaries and large and complex CIFs with many data items and many loops in a data block might require a version of *CIFtbx* with larger values of NUMDICT, NUMBLOCK and, perhaps, of NUMLOOP.

The variables NUMPAGE and NUMCPP control the amount of memory to be used to buffer the direct-access file and the size of the data transfers to and from that file. Smaller values will reduce the demand for memory at the expense of slower execution.

5.4.12.3. Use of INCLUDE

The INCLUDE statement allows the statements in the specified file to be treated as if they were being included in a program in place of the INCLUDE statement itself. This simplifies the maintenance of common-block declarations and is an important tool in keeping code well organized. In *CIFtbx*, the INCLUDE statement is used to bring the statements in the files ciftbx.cmn and ciftbx.sys into programs where they are needed, and to simplify ciftbx.cmn and ciftbx.sys by using INCLUDEs of the files ciftbx.cmv and ciftbx.cmf. The file ciftbx.cmv contains the definitions of the essential *CIFtbx* data structures as common blocks, for inclusion in both ciftbx.cmn for user applications and in ciftbx.sys for the *CIFtbx* library routines themselves. Most compilers handle the INCLUDE statement, but, if necessary, a user may replace any or all of the INCLUDE statements with the contents of the indicated file. For example, the only non-comments in ciftbx.cmn are

Table 5.4.12.1. *Parameter statements in CIFtbx*

NUMCHAR	Maximum number of characters in data names (default 48)
MAXBUF	Maximum number of characters in a line (default 200, increased to 4096 in *CIFtbx* 2.7 and later)
NUMPAGE	Number of memory resident pages (default 10)
NUMCPP	Number of characters per page (default 16 384)
NUMDICT	Number of entries in dictionary tables (default 3200)
NUMHASH	Number of hash table entries (a modest prime, default 53)
NUMBLOCK	Number of entries in data-block tables (default 500)
NUMLOOP	Number of loops in a data block (default 50)
NUMITEM	Number of items in a loop (default 50)
MAXTAB	Maximum number of tabs in output CIF line (default 10)
MAXBOOK	Maximum number of simultaneous bookmarks (default 1000)

```
include  'ciftbx.cmv'
include  'ciftbx.cmf'
```

This means that the file ciftbx.cmn could be replaced by a concatenation of the two files ciftbx.cmv and ciftbx.cmf.

5.4.12.4. Use of ENDDO

CIFtbx makes some use of the ENDDO statement (as well as nested IF, THEN, ELSE, ENDIF constructs) to improve readability of the source code. Most compilers accept the ENDDO statement, but if conversion is needed then constructs of the form

```
      do index = istart, iend, incr
        ...
      enddo
```

should be changed to

```
      do nnn index = istart, iend, incr
        ...
nnn   continue
```

where *nnn* is a unique statement number, not used elsewhere in the same routine.

5.4.12.5. Names of internal routines

The following routines are used internally by later versions of *CIFtbx*. If these names are needed for other routines, then changes in the library will be needed to avoid conflicts.

(*a*) Variable initialization:

```
block data
```

Critical *CIFtbx* variables are initialized with data statements in a block data routine.

(*b*) Control of floating-point exceptions:

```
subroutine clearfp
```

If a system requires special handling of floating-point exceptions, the necessary calls should be added to this subroutine.

(*c*) Message processing:

```
subroutine err (mess)
  character mess*(*)
subroutine warn (mess)
  character mess*(*)
subroutine cifmsg (flag, mess)
  character mess*(*), flag*(*)
```

Error and warning messages are processed through these three routines.

(*d*) Internal service routines:

```
subroutine dcheck
  (name, type, flag, tflag)
  logical flag, tflag
  character name*(*), type*4
subroutine eotext
subroutine eoloop
subroutine excat
  (sfname, bcname, lbcname)
  character*(*) sfname, bcname
  integer lbcname
subroutine getitm (name)
  character name*(*)
subroutine getstr
subroutine getlin (flag)
  character flag*4
subroutine putstr (string)
  character string*(*)
```

These routines are used internally by the library. The subroutine `dcheck` validates names against dictionaries. The subroutines `eotext` and `eoloop` are used to ensure termination of loops and text strings. The subroutines `getitm`, `getstr` and `getlin` extract items, strings and lines from the input CIF. The subroutine `putstr` writes strings to the output CIF.

(*e*) Numeric routines:

```
subroutine ctonum
subroutine putnum (numb, sdev, prec)
  double precision numb, sdev, prec
```

The routine `ctonum` converts a string to a number and its standard uncertainty. The subroutine `putnum` converts a number and standard uncertainty to an output string.

(*f*) String manipulation:

```
subroutine detab
integer function lastnb (str)
  character str*(*)
character*(MAXBUF) function locase (name)
  character name*(*)
```

The subroutine `detab` converts tabs to blanks. The function `lastnb` finds the column position of the last non-blank character in a string. The function `locase` converts a string to lower case.

(*g*) Hash-table processing:

```
subroutine hash_find (name, name_list,
  chain_list, list_length, num_list,
  hash_table, hash_length, ifind)
  character name*(*),
    name_list(list_length)
  integer hash_length,
    chain_list(list_length),
    hash_table(hash_length), ifind
subroutine hash_store (name, name_list,
  chain_list, list_length, num_list,
  hash_table, hash_length, ifind)
  character name*(*),
    name_list(list_length)
  integer hash_length,
    chain_list(list_length),
    hash_table(hash_length), ifind
integer function hash_value (name, hash_length)
  character name*(*)
  integer hash_length
```

These routines are used to manipulate the internal hash tables used by the library.

5.4.12.6. Use of the underscore character

All the externally accessible *CIFtbx* commands and variables terminate with the underscore character. This works well on most systems, but can cause occasional problems, because traditional Fortran does not include the underscore in the character set and some operating systems reserve the underscore as a system flag, for example to distinguish C-language library routines from those written in Fortran. If conversion is needed, and the local compiler allows long variable and subroutine names, then the simplest approach would be to make a local variant of *CIFtbx* in which every occurrence of underscore in a function, subroutine or variable name is changed to a distinctive character pattern (*e.g.* 'CIF' or 'qq'), but caution is needed, since there are many *character string*s used in the library that include the underscore. For example, in changing the variable `loop_` to loopCIF, it would be a mistake to change the statement

```
if(strg_(1:5).eq.'loop_')
  type_='loop'
```

to

```
if(strg_(1:5).eq.'loopCIF')
  type_='loop'
```

5.4.12.7. Names longer than six characters

CIFtbx uses some function, subroutine and variable names longer than six characters to improve readability, but, in most cases, consistent truncation of all uses of a name to six characters will not cause any problems.

5.4.12.8. File management

CIFtbx allows the user to read from one CIF while writing to another. The input CIF is first copied to a direct-access file to allow random access to desired portions of the input CIF. Since CIF allows data items to be presented in any order, the alternatives to the use of a direct-access file would have been to create memory-resident data structures for the entire CIF or to track position and make multiple search passes through the file as data items are requested. When programming for personal and laboratory computers with limited memory and which may lack virtual memory capabilities, assuming the availability of enough memory for large CIFs would greatly restrict the applications within which *CIFtbx* could be used. However, the disk accesses involved in using a direct-access file slow execution. When working on larger computers, execution speed can be increased at the expense of memory by increasing the number of memory-resident pages (see the parameter `NUMPAGE` above). If the number of pages times the number of characters per page (`NUMCPP`) is large enough to hold the entire CIF, the application will run much faster.

Direct reading of the input CIF, making multiple passes when data items are requested in a different order to that in which they are presented in the CIF, is only practical when the number of out-of-order requests is small and the applications will not need to be used as a filter, perhaps reading the output of another program 'on-the-fly'. Since we cannot predict the range of applications and CIFs for which *CIFtbx* will be used, and direct reading could become impossibly slow, *CIFtbx* uses a direct-access file.

The processing of an output CIF is simpler than reading a CIF. The application determines the order in which the writing is to be done. No sorting is normally needed. Therefore *CIFtbx* writes an output CIF directly.

5.4.12.9. Case sensitivity

A CIF may contain data names in upper, lower or a mixture of cases. Internally, *CIFtbx* does all its name comparisons in lower case, using the function `locase` (see above) to convert. Good style, however, dictates the use of certain case combinations in certain names. Therefore *CIFtbx* does this lower-case conversion as needed, preserving the original case for whatever use may be desired. An application needing maximum speed and which does not need to preserve the cases in the original CIF might consider doing the case conversion once and removing the use of `locase`.

5.4.12.10. Management of white space

CIF does not care about white space. One blank or tab is equivalent to many blanks or tabs or empty lines in separating data names from values and values from one another. The internal routine `getstr` extracts the next white-space-delimited string, using `getlin` to deliver input lines from the direct-access file as required. Since Fortran does not provide dynamic memory allocation, this approach presents a problem with multi-line text fields. Rather than allocate a large fixed space that might not hold still larger text fields, the library delivers those strings one line at a time. As with case sensitivity, *CIFtbx* does white-space scanning repeatedly, keeping the original presentation (including tabs) available should an application need access to it. The author of an application needing maximum speed, not needing the original presentation and wishing to conserve disk space might wish to modify the operation of *CIFtbx* to remove all comments and compress all separating white space to single blanks or line terminators in an initial sweep.

5.4.13. Distribution

Version 2.6.4 and an early release of version 3 of *CIFtbx* are included on the accompanying CD-ROM. As later versions are developed they will be available from the IUCr (http://www.iucr.org/iucr-top/cif) and authors' (http://www.bernstein-plus-sons.com/software/ciftbx) web sites.

The release kit is a compressed C-shell archive ciftbx.cshar.Z or a compressed shell archive ciftbx.shar.Z. Only one is needed. The uncompressed files ciftbx.cshar or ciftbx.shar are needed for implementation.

We are grateful to Frances C. Bernstein for her helpful comments and suggestions.

References

Bernstein, F. C. & Bernstein, H. J. (1996). *Translating mmCIF data into PDB entries. Acta Cryst.* A**52** (Suppl.), C576. Software available at http://www.bernstein-plus-sons.com/software/cif2pdb.

Bernstein, H. J. (1997). *cif2cif. CIF copy program*. Bernstein + Sons, Bellport, NY, USA. Included in http://www.bernstein-plus-sons.com/software/ciftbx.

Bernstein, H. J., Bernstein, F. C. & Bourne, P. E. (1998). *CIF applications. VIII. pdb2cif: translating PDB entries into mmCIF format. J. Appl. Cryst.* **31**, 282–295. Software available at http://www.bernstein-plus-sons.com/software/pdb2cif.

Bernstein, H. J. & Hall, S. R. (1998). *CIF applications. VII. CYCLOPS2: extending the validation of CIF data names. J. Appl. Cryst.* **31**, 278–281. Software available at http://www.bernstein-plus-sons.com/software/ciftbx/cyclops.src.

Bray, T., Paoli, J. & Sperberg-McQueen, C. M. (1998). *Extensible Markup Language (XML)*. W3C recommendation 10-February-1998. http://www.w3.org/TR/1998/REC-xml-19980210.

Hall, S. R. (1993*a*). *CIF applications. II. CIFIO: for CIF input/output in the Xtal system. J. Appl. Cryst.* **26**, 474–479.

Hall, S. R. (1993*b*). *CIF applications. IV. CIFtbx: a tool box for manipulating CIFs. J. Appl. Cryst.* **26**, 482–494.

Hall, S. R. & Bernstein, H. J. (1996). *CIF applications. V. CIFtbx2: extended tool box for manipulating CIFs. J. Appl. Cryst.* **29**, 598–603.

Keller, P. A. (1996). *A mmCIF toolbox for CCP4 applications*. *Acta Cryst.* A**52** (Suppl.), C576.

Murray-Rust, P. & Rzepa, H. (1999). *Chemical markup, XML and the Worldwide Web. 1. Basic principles. J. Chem. Inf. Comput. Sci.* **39**, 928–942.

5.5. The use of mmCIF architecture for PDB data management

BY J. D. WESTBROOK, H. YANG, Z. FENG AND H. M. BERMAN

5.5.1. Introduction

The Protein Data Bank (PDB) is an archive for macromolecular structures (Bernstein *et al.*, 1977; Berman *et al.*, 2000) and a major component of a global resource for macromolecular structural science (Berman *et al.*, 2003). The scale of its data handling operations is large, and depends on the effective exploitation of the latest developments in the science and technology of informatics. A significant component of its data storage and retrieval strategy is the management of structural data in mmCIF format with appropriate extensions.

Over its 30-year history, the PDB archive has grown from seven entries in 1973 to a collection of over 30 000 structures as of May 2005. The growth in the size of the archive has been accompanied by increases in both data content and in the structural complexity of individual entries. As the PDB has grown, there has been a significant broadening of its user community. In response to this change, the role of the PDB has expanded from being simply a provider of structure data files to providing a key information resource for the structural biology community.

Looking forward, an acceleration in the growth of the PDB archive is anticipated owing to developments in high-throughput structural determination methodologies and worldwide structural genomics efforts. To support the continued growth and evolution of the PDB archive, a framework is required that supports automation and scalability, and that can adapt to changes in both data content and delivery technology.

At the core of the PDB informatics infrastructure is an ontology of data definitions which electronically encode domain information in the form of precise definitions, examples and controlled vocabularies. In addition to domain information, data definitions also encode information such as data type, data relationships, range restrictions and presentation units.

The software-accessible PDB exchange data dictionary (Appendix 3.6.2) is the key part of the PDB informatics infrastructure. The exchange dictionary is an extension of the macromolecular Crystallographic Information File (mmCIF) data dictionary (Bourne *et al.*, 1997). The dictionary provides the foundation for software tools which exchange and validate data, create and load databases, translate data formats, and serve application program interfaces. The components of the informatics infrastructure developed by the PDB are being used to build a data pipeline to support high-throughput structure determination.

5.5.2. Representing macromolecular structure data

Macromolecular structure data have historically been represented in a simple record-oriented format developed by the PDB; this format has been widely used in structural and computational biology.

Affiliations: JOHN D. WESTBROOK, HUANWANG YANG, ZUKANG FENG and HELEN M. BERMAN, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA.

```
CRYST1  129.230   60.440   56.630  90.00 119.05  90.00 C 1 2 1       4
ATOM      1  N   ASP A   1      23.482  -0.621  -1.419  1.00 35.27           N
ATOM      2  CA  ASP A   1      24.897  -0.728  -1.885  1.00 32.46           C
ATOM      3  C   ASP A   1      25.573   0.515  -1.339  1.00 28.22           C
ATOM      4  O   ASP A   1      24.918   1.359  -0.744  1.00 29.11           O
ATOM      5  CB  ASP A   1      24.976  -0.729  -3.427  1.00 38.24           C
```

Fig. 5.5.2.1. Excerpt of records from a PDB data file.

While this PDB format has in general been adequate for representing coordinate data, it has proved less satisfactory for the description of related information such as chemical and biological features and experimental methodology. To provide a more rigorous data encoding that includes all of this related information, the Protein Data Bank has in recent years adopted a comprehensive ontology of structure and experiment based on the content of the mmCIF data dictionary.

5.5.2.1. PDB format

For the past 30 years, the PDB has served as the single central repository for macromolecular structure data. The data format used to store archival entries in the PDB is a column-oriented data format resembling many data formats developed to accommodate the limitations of paper punched-card technology (see Chapter 1.1). An example of the data format is shown in Fig. 5.5.2.1.

Many of the data records in this format are prefixed with a record tag (*e.g.* CRYST1, ATOM) followed by individual items of data. The specifications for the records in this data format are described informally by Callaway *et al.* (1996). In addition to the labelled records as in Fig. 5.5.2.1, many data records in the PDB format are presented as unstructured or only semi-structured remark records.

5.5.2.2. Ontology representation of macromolecular structure data

In 1998, the Research Collaboratory for Structural Bioinformatics (RCSB) assumed the management responsibilities for the PDB. One important outcome was the change in the underlying data representation used to process PDB data. The PDB now collects and processes data using a data representation based on a comprehensive ontology of macromolecular structure and experiment: the PDB exchange data dictionary. This representation is an extension of the mmCIF data dictionary, now the standard data representation for experimentally determined three-dimensional macromolecular structures. The dictionary and data files based on this data ontology (Westbrook & Bourne, 2000) are expressed using Self-defining Text Archival and Retrieval (STAR) syntax (Chapter 2.1).

Although the mmCIF dictionary was developed within the crystallographic community, the metadata model employed by mmCIF is quite general and has been adopted by other application domains including NMR, molecular modelling and molecular recognition (dictionaries are available at http://mmcif.pdb.org/). Within the crystallographic community, metadata dictionaries have also been

developed for other types of diffraction experiments, electron-microscopy data and for the general description of image data. The metadata concepts and tools that have been developed to support mmCIF are sufficiently general that they may be applied to the description of data in virtually any application.

The demands of structural genomics projects have driven the development of extensions to capture an increased level of experimental detail. These are available at http://mmcif.pdb.org/. Extensions have also been introduced to describe NMR, cryo-electron microscopy and all aspects of protein production. The ability to rapidly add extensions and incorporate these into the PDB data-processing system is an important feature for supporting the rapidly evolving technologies associated with high-throughput structure determinations.

The mmCIF metadata architecture is built from three levels as illustrated in Fig. 5.5.2.2 (see also Chapter 2.6). Individual data files are described at the top level (*e.g.* Fig. 5.5.2.2*a*). The contents of these data files are defined by a data dictionary (*e.g.* Fig. 5.5.2.2*b*) in the next lower level (see Chapters 3.6 and 4.5). The attributes used in this data dictionary to build data definitions are in turn defined in the dictionary description language (DDL) (*e.g.* Fig. 5.5.2.2*c*) in the lowest level (see Chapters 2.6 and 4.10).

The major syntactical constructs used by mmCIF are illustrated in the data file example of Fig. 5.5.2.2(*a*). Each data item or group of data items is preceded by an identifying keyword. Groups of related data items are organized into data categories. Two categories, CELL and ENTITY_POLY_SEQ, are shown in the example. CELL contains an individual instance describing a single set of crystallographic cell constants. ENTITY_POLY_SEQ contains a `loop_` (*i.e.* table) of instances describing a polymer residue sequence. Essentially all mmCIF data are described as a set of tabular data structures.

Each mmCIF data item is defined in a data dictionary. Data definitions are given between save-frame delimiters (*i.e.* `save_`); apart from this, the data definitions share the same simple syntax as used in data files. An example definition for a crystallographic cell constant is shown in Fig. 5.5.2.2(*b*). Many features of the cell constant are described in this definition, including data type, range restrictions, units of expression, dependent quantities, related definitions, necessity and related precision estimate. Although not shown in this example, dictionary definitions can also include parent–child relationships that have important consequences in maintaining data consistency.

The attributes of each data definition are defined in the DDL dictionary. Fig. 5.5.2.2(*c*) shows example DDL definitions describing data types. DDL definitions have the same syntax as definitions used in the data dictionary. Because the attributes of the DDL are also used in DDL definitions, this metadata architecture is described as self-defining.

The RCSB PDB distributes parsing tools that support all three levels of this metadata architecture (http://sw-tools.pdb.org/). The *CIFPARSE_OBJ* package (Tosic & Westbrook, 2000) provides high-level methods to read, write, validate and manage data from data files, dictionaries and DDLs. Data files can be validated relative to an input data dictionary, and dictionary files can be validated relative to an input DDL. *CIFPARSE_OBJ* stores information in a collection of table objects. Access methods are provided to search and manipulate the table objects. A companion package, *CIFOBJ* (Schirripa & Westbrook, 1996), provides an alternative representation of dictionary and DDL data. *CIFOBJ* organizes dictionary information into a collection of category and item-level objects. Access methods are provided for all dictionary attributes.

```
_cell.entry_id      W1QQQ
_cell.length_a      129.230
_cell.length_b       60.440
_cell.length_c       56.630
_cell.angle_alpha    90.00
_cell.angle_beta    119.05
_cell.angle_gamma    90.00
_cell.Z_PDB           4
loop_
  _entity_poly_seq.entity_id
  _entity_poly_seq.num
  _entity_poly_seq.mon_id
    1    1     ASP       1    2     ILE
    1    3     VAL       1    4     LEU
    1    5     THR       1    6     GLN
    1    7     SER       1    8     PRO
    1    9     ALA       1    10    SER
```

(*a*)

```
save__cell.length_a
    _item_description.description
; Unit-cell length a corresponding to the structure
  reported.
;
    _item.name                      '_cell.length_a'
    _item.category_id                 cell
    _item.mandatory_code              no
    _item_aliases.alias_name        '_cell_length_a'
    _item_aliases.dictionary          cif_core.dic
    _item_aliases.version             2.0.1
     loop_
    _item_dependent.dependent_name
                                    '_cell.length_b'
                                    '_cell.length_c'
     loop_
    _item_range.maximum
    _item_range.minimum              .    0.0
                                    0.0   0.0
    _item_related.related_name      '_cell.length_a_esd'
    _item_related.function_code       associated_esd
    _item_sub_category.id             cell_length
    _item_type.code                   float
    _item_type_conditions.code        esd
    _item_units.code                  angstroms
    save_
```

(*b*)

```
save_ITEM_TYPE_LIST
    _category.description
;   Attributes which define each type code.
;
    _category.id                    item_type_list
    _category.mandatory_code        no
    _category_key.name            '_item_type_list.code'
     loop_
    _category_group.id              'ddl_group'
                                    'item_group'
     save_

save__item_type_list.code
    _item_description.description
;    The codes specifying the nature of the data value.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_item_type_list.code'   item_type_list      yes
         '_item_type.code'        item_type           yes

    _item_type.code                 code
    _item_linked.child_name       '_item_type.code'
    _item_linked.parent_name      '_item_type_list.code'

     save_
```

(*c*)

Fig. 5.5.2.2. Files at different levels of the mmCIF metadata architecture. (*a*) mmCIF data file excerpt. (*b*) Example mmCIF data dictionary definition. (*c*) Example DDL dictionary attribute definition.

5.5.2.3. Supporting other data formats and data delivery methods

One of the greatest benefits of a dictionary-based informatics infrastructure is the flexibility that it provides in supporting alternative data formats and delivery methods. Because the data and all of their defining attributes are electronically encoded, translation between data and dictionary formats can be achieved using lightweight software filters without loss of any information.

XML provides a particularly good example of the ease with which data can be converted to and from the mmCIF format. XML translations of mmCIF data files are currently provided on the RCSB PDB beta ftp site (ftp://beta.rcsb.org/pub/pdb/uniformity/data/XML/). These XML files use mmCIF dictionary data-item names as XML tags. These files were created by a translation tool (http://sw-tools.pdb.org/apps/MMCIF-XML-UTIL/) that translates mmCIF data files to XML in compliance with an XML schema. The XML schema is similarly software-translated from the PDB exchange data dictionary.

Other delivery methods such as Corba (http://www.omg.org/cgi-bin/doc?lifesci/00-02-02) do not require a data format, as data are exchanged using an application program interface (API). A Corba API for macromolecular structure (Greer *et al.*, 2002) based on the content of the mmCIF data dictionary has been approved by the Object Management Group (OMG). Software tools supporting this Corba API (*OpenMMS*, http://openmms.sdsc.edu, and *FILM*, http://sw-tools.pdb.org/apps/FILM) take full advantage of the data dictionary in building the interface definitions and supporting server on which the API is based (see also Section 5.3.8.2).

5.5.3. Integrated data-processing system: overview

The RCSB PDB data-processing system has been designed to take full advantage of the features of the mmCIF metadata framework. The AutoDep Input Tool (*ADIT*) is an integrated data-processing system developed to support deposition, data processing and annotation of three-dimensional macromolecular structure data.

This system, which is outlined in Fig. 5.5.3.1, accepts experimental and structural data from a user for deposition. Data are input in the form of data files or through a web-based form interface. The input data can be validated in a very basic sense for syntax compliance and internal consistency. Other computational validation can also be applied, including checking the input structure data against a variety of community standard geometrical criteria and comparing the input experimental data with the derived structure model. The suite of validation software used within *ADIT* is distributed separately (http://sw-tools.pdb.org/apps/VAL/). All of this validation information is returned to the user as a collection of HTML reports.

In addition to providing data-validation reports, *ADIT* also encodes data in archival data files and loads data into a relational database. The loading of data into the relational database is aided by an expert annotator. The *ADIT* system customizes its behaviour according to the user's requirements. One important distinction is between the behaviour of the interface provided for depositing data and that of the interface used for annotating the data. The depositor is focused only on data collection and provides the simplest possible presentation of the information to be input. The annotator sees the detail of all possible data items as well as the full functionality of the supporting data-processing software and database system.

Although the *ADIT* system was originally developed to support the centralized data deposition and annotation of macromolecular structure data, it is not limited to these particular applications. Because the architecture of the *ADIT* system derives the full scope of information to be processed from a data dictionary, the system can transparently provide data input and processing functionality for any content domain. This feature has been exploited in building a data-input tool for the BioSync project (Kuller *et al.*, 2002). The *ADIT* system can also be configured in workstation mode to provide single-user data collection and processing functionality. This version of the *ADIT* system as well as the supporting mmCIF parsing and data-management tools are currently distributed by the RCSB PDB under an open-source licence (http://sw-tools.pdb.org/apps/ADIT).

5.5.3.1. *ADIT*: functional description

The basic functions of the *ADIT* deposition system are shown in Fig. 5.5.3.2. Users interact with the *ADIT* system through a web server. The CGI components of the *ADIT* system (that is, functional software components interacting with web input data through the Common Gateway Interface protocol) dynamically build the HTML that provides the system user interface. These CGI components are currently implemented as compiled binaries from C++ source code.

User data can be provided in the form of data files or as keyboard input. Input files can be accepted in a variety of formats. *ADIT* uses a collection of format filters to convert input data to the data specification defined in a persistent data dictionary. Data in the form of data files are typically loaded first. Any input data that are not included in uploaded files can be keyed in by the user. *ADIT* builds a set of HTML forms for each category of data to be input. At any point during an input session, a user may choose to view or deposit the input data. Users who are depositing data may also use the data-validation services through the *ADIT* interface.

Comprehensive data ontologies like the PDB exchange dictionary contain vast numbers of data definitions. A data-input application may only need to access a small fraction of these definitions at any point. To address the problem of selecting only the relevant set of input data items from a data dictionary *ADIT* uses a view database. In addition to defining the scope of the data items to be edited by the *ADIT* application, an *ADIT* data view also stores

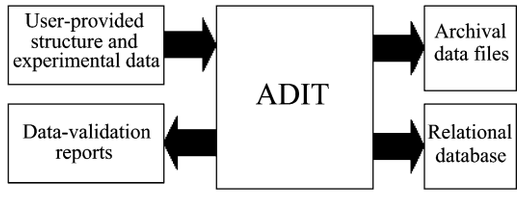


Fig. 5.5.3.1. Functional diagram of the *ADIT* system.

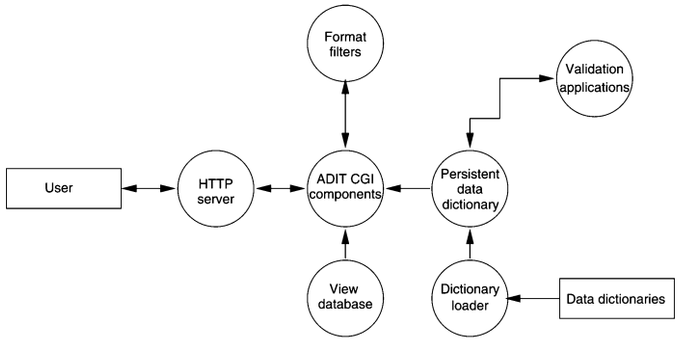


Fig. 5.5.3.2. Schematic diagram of *ADIT* editing, format translation and validation functions.

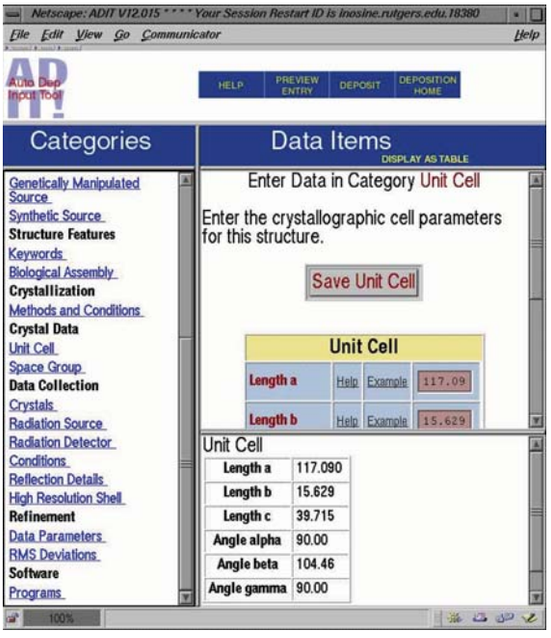


Fig. 5.5.3.3. Example *ADIT* data-input screen.

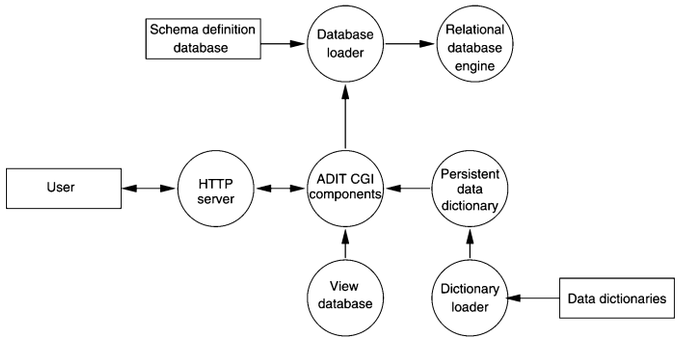


Fig. 5.5.3.4. Schematic diagram of *ADIT* database loading functions.

presentation details that are used in building the HTML input forms. An important use of the data view is to provide a simple and intuitive presentation of information for novice users which disguises the complex details of a data dictionary.

Fig. 5.5.3.3 shows an example *ADIT* editing screen for the crystallographic unit cell. The data dictionary category containing this information is named CELL, and the length of the first cell axis is defined in the dictionary as `_cell.length_a` (Fig. 5.5.2.2*b*). In this case, the data view has substituted *Unit Cell* and *Length a* for the dictionary data names. Although this example is simple, some dictionary data names are as long as 75 characters, and in these instances the ability to display a simpler name is essential.

Precise dictionary definitions and examples obtained from the data dictionary are accessible from the *ADIT* interface through buttons next to each data item. *ADIT* makes full use of the dictionary specification in data-input operations. Data items defined to assume only specific values have pulldown menus or selection boxes. Data type and range restrictions are checked when data are input and diagnostics are displayed to the user if errors are detected.

For performance reasons, the data dictionary is converted from its tabular text structure to an object representation using *CIFOBJ*. The class supporting the object representation provides efficient access functions to all of the data dictionary attributes. A dictionary loader is used to check the consistency of the data dictionary and to load the object representation from the text form of the data dictionary.

Any dictionary that complies with the dictionary description language (DDL2) can be loaded and used by *ADIT*. All *ADIT* software components gain their knowledge of the input data from the data dictionary and any associated data views. Consequently, *ADIT* can be tailored for use in virtually any data-input and data-processing application.

5.5.3.2. Generalized database support

In addition to the data editing and processing functions, *ADIT* also supports a versatile database loader (*mmCIF Loader*; http://sw-tools.pdb.org/apps/MMCIF-LOADER) that builds database schemata and extracts the processed data required to load database instances. The relation of the database loader to the central components of the *ADIT* system is shown in Fig. 5.5.3.4.

Schemata are defined in a metadata repository that is accessed by the loader application. In the simplest case, a schema can be constructed that is modelled directly from the data dictionary. Since the data model underlying the dictionary description language used to build *ADIT* data dictionaries is essentially relational, mapping a data dictionary specification to a relational schema is straightforward.

In other cases, a mapping is required between the target schema and the data dictionary specification. This mapping is encoded in the schema metadata repository. The database loader uses this mapping information to extract items from data files and translate these data into a form that can be loaded into the target database schema. The definition of the mapping operation can include: selection operations with equijoin constraints (*e.g.* the value of `_entity.type` where `_entity.id` = 1), aggregation (*e.g.* count, sum, average), collapse (*e.g.* vector to string), type conversions and existence tests.

Schema definitions are converted by the database loader into SQL instructions that create the defined tables and indices. Loadable data are produced either as SQL insert/update instructions or in the more efficient table copy formats used by popular database engines (*i.e.* DB2, Sybase, Oracle and MySQL). Loadable data can also be produced in XML.

5.5.3.3. Building a structure-determination data pipeline

One goal of high-throughput structural genomics is the automatic capture of all the details of each step in the process of structure determination. Fig. 5.5.3.5 shows a simplified structure-determination data pipeline. The essential details of each pipeline step are extracted and later assembled to make a data file for PDB deposition. The RCSB PDB data-processing infrastructure has been developed in anticipation of a data pipeline in which automated deposition would be the terminal step. The dictionary technology and software tools developed by the RCSB PDB to process and manage mmCIF data can be reused to provide the data-handling operations required to build the pipeline.

Dictionary definitions have been carefully developed to describe the details of each step in the structure-determination pipeline. These data items are typically accessible in electronic form after each program step. The information is either exported directly in mmCIF format or is printed in a program output file. To deal with the latter case, a utility program, *PDB_EXTRACT* (http://sw-tools.pdb.org/apps/PDB_EXTRACT), has been developed to parse program output files and extract key data values. In either case, the results of this incremental extraction of data from each program step must be merged to build a complete mmCIF

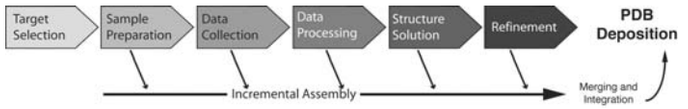


Fig. 5.5.3.5. Schematic diagram of a structure-determination data pipeline.

data file ready for deposition. The *PDB_EXTRACT* program also carrys out this merging operation.

Some steps in the structure-determination pipeline may not be driven by software. For instance, the details of protein production may be held in laboratory databases or within laboratory notebooks. A version of *ADIT* with a data view including all of the structural genomics data extensions has been created for entering these data. This *ADIT* tool can also be used to validate and check the completeness of the final data file.

5.5.4. Access

All of the software tools and libraries described in this chapter are distributed with full source under an open-source licence. Applications are also distributed in binary form for Intel/Linux, Sun/Solaris, SGI/IRIX and Dec Alpha platforms. Several are also included on the CD-ROM accompanying this volume.

The RCSB/PDB is operated by Rutgers, The State University of New Jersey; the San Diego Supercomputer Center at the University of California, San Diego; and the Center for Advanced Research in Biotechnology of the National Institute of Standards and Technology. RCSB/PDB is supported by funds from the National Science Foundation (NSF), the National Institute of General Medical Sciences (NIGMS), the Department of Energy (DOE), the National Library of Medicine (NLM), the National Cancer Institute (NCI), the National Center for Research Resources (NCRR), the National Institute of Biomedical Imaging and Bioengineering (NIBIB), and the National Institute of Neurological Disorders and Stroke (NINDS).

References

Berman, H. M., Henrick, K. & Nakamura, H. (2003). *Announcing the worldwide Protein Data Bank. Nature Struct. Biol.* **10**, 980.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *The Protein Data Bank. Nucleic Acids Res.* **28**, 235–242.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F. Jr, Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *The Protein Data Bank: a computer-based archival file for macromolecular structures. J. Mol. Biol.* **112**, 535–542.

Bourne, P. E., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. & Fitzgerald, P. M. D. (1997). *Macromolecular Crystallographic Information File. Methods Enzymol.* **277**, 571–590.

Callaway, J., Cummings, M., Deroski, B., Esposito, P., Forman, A., Langdon, P., Libeson, M., McCarthy, J., Sikora, J., Xue, D., Abola, E., Bernstein, F., Manning, N., Shea, R., Stampf, D. & Sussman, J. (1996). *Protein Data Bank contents guide: Atomic coordinate entry format description.* Brookhaven National Laboratory, New York, USA. Available from http://www.rcsb.org/pdb/docs/format/pdbguide2.2/part_0.html.

Greer, D. S., Westbrook, J. D. & Bourne, P. E. (2002). *An ontology driven architecture for derived representations of macromolecular structure. Bioinformatics*, **18**, 1280–1281.

Kuller, A., Fleri, W., Bluhm, W. F., Smith, J. L., Westbrook, J. & Bourne, P. E. (2002). *A biologist's guide to synchrotron facilities: the BioSync web resource. Trends Biochem. Sci.* **27**, 213–215.

Schirripa, S. & Westbrook, J. D. (1996). *CIFOBJ. A class library of mmCIF access tools.* Reference guide. http://sw-tools.pdb.org/apps/CIFOBJ/cifobj/index.html.

Tosic, O. & Westbrook, J. D. (2000). *CIFParse. A library of access tools for mmCIF.* Reference guide. http://sw-tools.pdb.org/apps/CIFPARSE-OBJ/cifparse/index.html.

Westbrook, J. & Bourne, P. E. (2000). *STAR/mmCIF: an ontology for macromolecular structure. Bioinformatics*, **16**, 159–168.

5.6. *CBFlib*: an ANSI C library for manipulating image data

BY P. J. ELLIS AND H. J. BERNSTEIN

5.6.1. Introduction

CBFlib is a library of ANSI C functions providing a simple mechanism for accessing crystallographic binary files (CBFs) and image-supporting CIF (imgCIF) files (see Chapters 2.3, 3.7 and 4.6). The *CBFlib* application programming interface (API) consists of a set of low-level functions and a set of high-level functions. The low-level functions are loosely based on the *CIFPARSE* (Tosic & Westbrook, 2000) API for mmCIF files. As in *CIFPARSE*, the low-level functions in *CBFlib* do not perform any semantic integrity checks and simply create, read, modify and write files conforming to the CIF syntax, with additional functionality for working with binary sections. These basic functions are independent of the details of the CBF/imgCIF dictionary and can be used to manage any CIF data set. In contrast, the high-level functions are based on the CBF/imgCIF dictionary and facilitate writing or reading commonly used entries to or from CBF and imgCIF data files.

External to a program, a CBF/imgCIF data set 'lives' in a file. Internally, when managed by *CBFlib*, a CBF or imgCIF data set has a simple tree structure pointed to by a 'handle' (Fig. 5.6.1.1). At the highest level are named data blocks. Each data block may contain a number of named categories. Within each category, the actual data entries are stored in tabular form with named columns and numbered rows. The numbers of rows in different columns of a given category are constrained by the software to be the same.

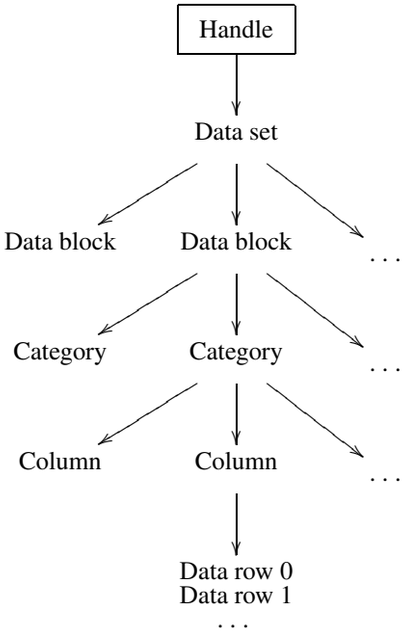


Fig. 5.6.1.1. Schematic data structure of a CBF or imgCIF data set accessed through a handle.

Table 5.6.1.1. *Return values from CBFlib functions*

Error code	Hexadecimal value	Meaning
`CBF_FORMAT`	1	Invalid file format
`CBF_ALLOC`	2	Memory allocation failed
`CBF_ARGUMENT`	4	Invalid function argument
`CBF_ASCII`	8	Value is ASCII (not binary)
`CBF_BINARY`	10	Value is binary (not ASCII)
`CBF_BITCOUNT`	20	Expected number of bits does not match the actual number written
`CBF_ENDOFDATA`	40	End of the data reached before end of array
`CBF_FILECLOSE`	80	File close error
`CBF_FILEOPEN`	100	File open error
`CBF_FILEREAD`	200	File read error
`CBF_FILESEEK`	400	File seek error
`CBF_FILETELL`	800	File tell error
`CBF_FILEWRITE`	1000	File write error
`CBF_IDENTICAL`	2000	A data block with the new name already exists
`CBF_NOTFOUND`	4000	Data block, category, column or row does not exist
`CBF_OVERFLOW`	8000	Number read cannot fit into the destination argument; destination has been set to the nearest value
`CBF_UNDEFINED`	10000	Requested number not defined
`CBF_NOTIMPLEMENTED`	20000	Requested functionality is not yet implemented

CBFlib provides functions to create a corresponding data structure in memory; to copy a data set from an external file to the data structure or from the data structure to an external file; to navigate the tree; to scan, add and remove data blocks within data sets, categories (tables) within data blocks, and rows or columns within categories; to read or modify data entries; and finally to delete the structure from memory.

As is common in C programming, all functions return an integer equal to 0 for success or an error code for failure. The *CBFlib* error codes are given in Table 5.6.1.1.

CBFlib is thread-safe, re-entrant and able to operate on multiple data sets at a time. This means that the library maintains no static data and that the object to be operated on must be passed to each function. In *CBFlib*, this is accomplished by referring to each data set in memory with a unique handle of type `cbf_handle`. The handle maintains a link to the data structure in memory as well as the current location on the tree (data block, category, column and row). Before reading or creating a data set, the handle is created by calling the `cbf_make_handle` function. When the data set is no longer required, the resources associated with it can be freed using `cbf_free_handle`. Most functions in the library expect a handle as the first argument.

CBF binary data files and imgCIF ASCII data files may have one or more large images or other data sections as values for CIF tags. The focus of *CBFlib* is to handle large data sections efficiently.

The basic flow of an application reading CBF/imgCIF data with the low-level *CBFlib* functions is shown in Fig. 5.6.1.2.

Affiliations: PAUL J. ELLIS, Stanford Linear Accelerator Center, 2575 Sand Hill Road, Menlo Park, CA 94025, USA; HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA.

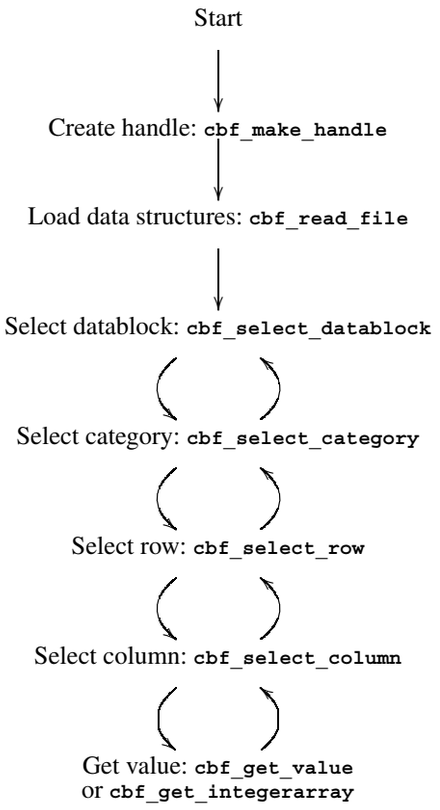


Fig. 5.6.1.2. Flow chart for a typical application reading CBF/imgCIF data.

The general approach to reading CBF/imgCIF data with *CBFlib* is to create an empty data structure with `cbf_make_handle`, load the data structures with `cbf_read_file` and then use nested loops to work through data blocks, categories, rows and columns in turn to extract values. Conceptually, all data values are held in the memory-resident data structures. In practice, however, only pointers to text fields with image data are held in memory. The data themselves remain on disk until explicitly referenced.

The basic flow of an application writing CBF/imgCIF data with the low-level *CBFlib* functions is shown in Fig. 5.6.1.3.

The general approach to writing CBF/imgCIF data with *CBFlib* is to create empty data structures with `cbf_make_handle` and load the data structures with nested loops, working through data blocks, categories, rows and columns in turn, to store values. The major difference from the nested loops used for reading is that empty columns are created before data are stored into the data structures row by row. Alternatively, the data could be stored column by column. Finally, the fully loaded memory data structures are written out with `cbf_write_file`. As with reading, text fields with image data are actually held on disk.

5.6.2. *CBFlib* function descriptions

All *CBFlib* functions have two common characteristics: (i) they return an integer equal to 0 for success or an error code for failure; (ii) any pointer argument for the result of an operation can be safely set to NULL. The error codes are given in Table 5.6.1.1.

CBFlib provides two low-level functions to create or destroy the structure used to hold a data set:

```
cbf_make_handle
cbf_free_handle
```

There are two functions to copy a data set from or into a file:

```
cbf_read_file
cbf_write_file
```

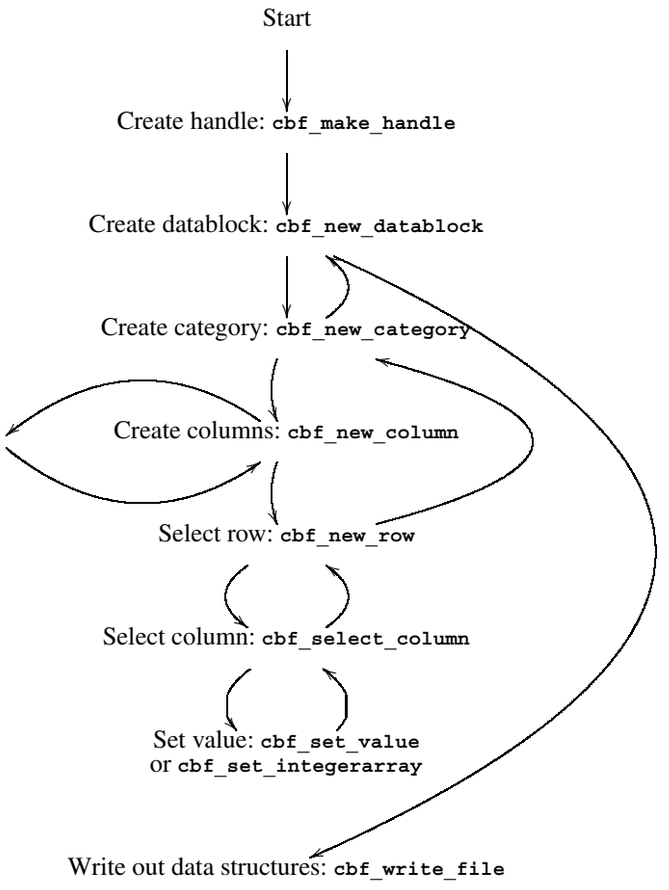


Fig. 5.6.1.3. Flow chart for an application writing CBF/imgCIF data.

The data structures 'behind' the handles retain pointers to current locations. This facilitates scanning through a CIF or CBF by data blocks, categories, rows and columns. The term '`rewind`' refers to setting the internal pointer for the type of item specified so that the first such item is pointed to.

In general, CIF does not permit duplication of the names of data blocks or category names. In practice, however, duplications do occur. *CBFlib* provides '`force`' variants of some functions to allow creation of duplicate names.

In *CBFlib*, the term '`set`' refers to changing the name of the currently specified item. The term '`reset`' refers to emptying a data block or category without deleting it. The term '`remove`' refers to deleting a data block, category, column or row. The terms '`select`' and '`next`' refer to finding the designated item by number, while the term '`find`' refers to finding the designated item by name.

CBFlib provides the following functions to manage data blocks and categories:

```
cbf_set_datablockname
cbf_ { new, force_new, reset, remove, rewind, select, next, find } { _datablock, _category }
cbf_reset_datablocks
cbf_count { _datablocks, _categories }
cbf_ { datablock, category } _name
```

The following functions manage columns and rows:

```
cbf_ { new | remove | rewind | next | find | count | select } { _column | _row }
cbf_column_name
cbf_row_number
cbf_ { insert | delete } _row
cbf_find_nextrow
```

The following functions are provided to manage data values:

```
cbf_ { get | set } { _value | _integervalue | _doublevalue | _integerarray }
cbf_get_integerarrayparameters
```

Two macro definitions are provided to facilitate the handling of errors:

```
cbf_failnez
cbf_onfailnez
```

CBFlib also provides higher-level routines to simplify the management of complex CBF/imgCIF data sets:

```
cbf_read_template
cbf_ { get | set } { _diffrn_id | _crystal_id | _wavelength | _polarization | _divergence | _gain | _overload | _integration_time | _time | _date | _image | _axis_setting }
cbf_count_elements
cbf_get_element_id
cbf_set_current_time
cbf_get_image_size
cbf_ { construct | free } { _goniometer | _detector }
cbf_get_rotation_ { axis | range }
cbf_rotate_vector
cbf_get_reciprocal
cbf_get_ { beam_center | detector_distance | detector_normal | pixel_coordinates | pixel_normals | pixel_area }
```

5.6.2.1. Low-level *CBFlib* functions

The prototypes for low-level *CBFlib* functions are defined in the header file cbf.h, which should be included in any program that uses *CBFlib*. As noted previously, every function returns an integer equal to 0 to indicate success or an error code on failure (Table 5.6.1.1).

Table 5.6.2.1. *Formal parameters for low-level CBFlib functions*

`array`	Untyped array, typically holding a pointer to an image
`binary_id`	Integer identifier of a binary section
`categories`	Integer used for a count of categories
`category`	Integer ordinal of a category, counting from 0
`categoryname`	Character string; the name of a category
`ciforcbf`	Integer; selects the format in which the binary sections are written (CIF/CBF)
`column`	Integer ordinal of a column, counting from 0
`columnname`	Character string; the name of a column
`columns`	Integer count of columns in a category
`compression`	Integer designating the compression method used
`datablock`	Integer ordinal of a data block, counting from 0
`datablockname`	Character string; the name of a data block
`datablocks`	Integer count of data blocks in a CBF/imgCIF data set
`elements`	Number of elements in the array
`elements_read`	Pointer to the destination number of elements actually read
`elsigned`	Set to nonzero if the destination array elements are signed
`elsize`	Size in bytes of each array element
`elunsigned`	Pointer to an integer; set to 1 if the elements can be read as unsigned integers
`encoding`	Integer; selects the type of encoding used for binary sections and the type of line termination in imgCIF files
`file`	File descriptor
`handle`	CBF handle
`headers`	Integer; controls/selects the type of header in CBF binary sections and message digest generation
`maxelement`	Integer; largest element
`minelement`	Integer; smallest element
`number`	Integer or double value
`readable`	Integer; if nonzero: this file is random-access and readable, and can be used as a buffer
`row`	Integer; row ordinal
`rows`	Integer; row count
`value`	Integer or double value

The arguments to *CBFlib* functions are based on a view of a CBF/imgCIF data set as a tree (Fig. 5.6.1.1). The root of the tree is the data set and is identified by a handle that points to the data structures representing that tree. The main branches of the tree are the data blocks, identified by name or by number. Within each data block, the tree branches into categories, each of which branches into columns. Categories and columns also are identified by name or by number. Within each column is an array of values, the rows of which are identified by number. The current data block, category, column and row are stored in the data structures of a data set.

The following function descriptions include the formal parameters. When a '$*$' appears before a formal parameter, it is a pointer to the relevant value, rather than the actual value. The formal parameters for the low-level *CBFlib* functions are given in Table 5.6.2.1.

Before working with a CBF (or CIF), it is necessary to create a handle. When work with the CBF is completed, the handle and associated data structures should be released:

```
int cbf_make_handle (cbf_handle *handle);
int cbf_free_handle (cbf_handle handle);
```

Normally, processing cannot continue if a handle is not created. Typical code to create a handle is:

```
#include "cbf.h"
cbf_handle cif;

if ( cbf_make_handle (&cif) ) {
   fprintf(stderr,
     "Failed to create handle for input_cif\n");
   exit(1);
 }
```

Table 5.6.2.2. *Values for headers in* `cbf_read_file`

`MSG_DIGEST`	Check that the digest of the binary section matches any header value. If the digests do not match, the call will return `CBF_FORMAT`. The evaluation and comparison is delayed (a 'lazy' evaluation) to ensure maximal processing efficiency. If an immediate evaluation is desired, see `MSG_DIGESTNOW` below.
`MSG_DIGESTNOW`	Check that the digest of the binary section matches any header value. If the digests do not match, the call will return `CBF_FORMAT`. This evaluation and comparison is performed during initial parsing of the section to ensure timely error reporting at the expense of processing efficiency. If a more efficient delayed ('lazy') evaluation is desired, see `MSG_DIGEST` above.
`MSG_NODIGEST`	Do not check the digest (default).

Once a handle has been created, the data structures can be loaded with all the information held in a CBF file:

```
int cbf_read_file (cbf_handle handle, FILE *file,
     int headers);
```

Conceptually, all data values are associated with the handle at the `cbf_read_file` call. In practice, however, only the non-binary data are actually stored in memory. To work with potentially large binary sections most efficiently, these are skipped until explicitly referenced. For this reason, *`file`* must be a random-access file opened in binary mode [`fopen (..., "rb")`] and must not be closed by the calling program. *CBFlib* will call `fclose` when the file is no longer required.

The *`headers`* parameter controls the handling of any message digests embedded in the binary sections (Table 5.6.2.2). A *`headers`* value of `MSG_DIGEST` will cause the code to compare the digest of the binary section with any header message digest value. To maximize processing efficiency, this comparison will be delayed until the binary section is actually read into memory or copied (a 'lazy' evaluation). If immediate evaluation is required, use `MSG_DIGESTNOW`. In either case, if the digests do not match, the function in which the evaluation is taking place will return the error `CBF_FORMAT`. To ignore any digests, use the *`headers`* value `MSG_NODIGEST`.

The `cbf_write_file` call writes out the data associated with a CBF handle:

```
int cbf_write_file (cbf_handle handle, FILE *file,
     int readable, int ciforcbf, int headers,
     int encoding);
```

This call has several options controlling whether binary sections are written unencoded (CBF) or encoded in ASCII to conform to the CIF syntax (imgCIF), the type of headers in the binary sections, and the type of ASCII encoding and line termination used. The acceptable values for *`ciforcbf`* are `CIF` for ASCII-encoded binary sections or `CBF` for unencoded binary sections. The *`headers`* parameter (Table 5.6.2.3) can take the value `MIME_HEADERS` to select MIME-type binary section headers or `MIME_NOHEADERS` for simple ASCII headers. The value `MSG_DIGEST` will generate digests for validation of the binary data and the value `MSG_NODIGEST` will skip digest evaluation. The header and digest flags may be combined using the logical OR operator.

Similarly, there are several combinable flags for the parameter *`encoding`* (Table 5.6.2.4). `ENC_BASE64` selects BASE64 encoding, `ENC_QP` selects quoted-printable encoding, and `ENC_BASE8`, `ENC_BASE10` and `ENC_BASE16` select octal, decimal and hexadecimal, respectively. `ENC_FORWARD` maps bytes to words forward (1234) for BASE8, BASE10 or BASE16 encoding and `ENC_BACKWARD` maps bytes to words backward (4321). Finally, `ENC_CRTERM` terminates lines with carriage return (CR) and `ENC_LFTERM` terminates lines with line feed (LF) (thus `ENC_CRTERM|ENC_LFTERM` will use CR LF).

Table 5.6.2.3. *Values for headers in* `cbf_write_file`

Values may be combined bit-wise.

`MIME_HEADERS`	Use MIME-type headers (default)
`MIME_NOHEADERS`	Use simple ASCII headers
`MSG_DIGEST`	Generate message digests for binary data validation
`MSG_NODIGEST`	Do not generate message digests (default)

CBFlib maintains temporary storage on disk as necessary for files to be written, so that *`file`* does not have to be random-access. However, if it is random-access and readable, resources can be conserved by setting *`readable`* nonzero.

The remaining low-level functions are involved in navigating the tree structure, creating and deleting data blocks, categories and table columns and rows, and retrieving or modifying data values.

The navigation functions are:

```
int cbf_find_datablock (cbf_handle handle,
     const char *datablockname);
int cbf_find_category (cbf_handle handle,
     const char *categoryname);
int cbf_find_column (cbf_handle handle,
     const char *columnname);
int cbf_find_row (cbf_handle handle,
     const char *value);
int cbf_find_nextrow (cbf_handle handle,
     const char *value);
int cbf_select_datablock (cbf_handle handle,
     unsigned int datablock);
int cbf_select_category (cbf_handle handle,
     unsigned int category);
int cbf_select_column (cbf_handle handle,
     unsigned int column);
int cbf_select_row (cbf_handle handle,
     unsigned int row);
int cbf_rewind_datablock (cbf_handle handle);
int cbf_rewind_category (cbf_handle handle);
int cbf_rewind_column (cbf_handle handle);
int cbf_rewind_row (cbf_handle handle);
int cbf_next_datablock (cbf_handle handle);
int cbf_next_category (cbf_handle handle);
int cbf_next_column (cbf_handle handle);
int cbf_next_row (cbf_handle handle);
```

The function `cbf_find_datablock` selects the first data block with name *`datablockname`* as the current data block. Similarly, `cbf_find_category` selects the category within the current data block with name *`categoryname`* and `cbf_find_column` selects the corresponding column within the current category. The function `cbf_find_row` differs slightly in that it selects the first row in the current column with the corresponding *`value`* and `cbf_find_nextrow` selects the row with the corresponding value following the current row. Note that selecting a new data block makes the current category, column and row undefined and that selecting a new category similarly makes the column and row undefined. In contrast, repositioning by column does not change the current row and repositioning by row does not change the current column.

The remaining functions navigate on the basis of the order of the data blocks, categories, columns and rows. Thus, `cbf_select_datablock` selects data-block number *`datablock`*, counting from 0, `cbf_rewind_datablock` selects the first data

Table 5.6.2.4. *Values for encodings in* `cbf_write_file`

Values may be combined bit-wise.

`ENC_BASE64`	Use BASE64 encoding (default)
`ENC_QP`	Use quoted-printable encoding
`ENC_BASE8`	Use BASE8 (octal) encoding
`ENC_BASE10`	Use BASE10 (decimal) encoding
`ENC_BASE16`	Use BASE16 (hexadecimal) encoding
`ENC_FORWARD`	For BASE8, BASE10 or BASE16 encoding, map bytes to words forward (1234) (default on little-endian machines)
`ENC_BACKWARD`	For BASE8, BASE10 or BASE16 encoding, map bytes to words backward (4321) (default on big-endian machines)
`ENC_CRTERM`	Terminate lines with CR
`ENC_LFTERM`	Terminate lines with LF (default)

block and `cbf_next_datablock` selects the data block following the current data block.

All of these functions return `CBF_NOTFOUND` if the requested object does not exist.

The '`count`' functions evaluate the number of data blocks in the data set, the number of categories in the current data block and the number of columns or rows in the current category:

```
int cbf_count_datablocks (cbf_handle handle,
     unsigned int *datablocks);
int cbf_count_categories (cbf_handle handle,
     unsigned int *categories);
int cbf_count_columns (cbf_handle handle,
     unsigned int *columns);
int cbf_count_rows (cbf_handle handle,
     unsigned int *rows);
```

The '`name`' functions retrieve the current data block, category or column names:

```
int cbf_datablock_name (cbf_handle handle,
     const char **datablockname);
int cbf_set_datablockname (cbf_handle handle,
     const char *datablockname);
int cbf_category_name (cbf_handle handle,
     const char **categoryname);
```

As rows do not have names, the corresponding function is:

```
int cbf_row_number (cbf_handle handle,
     unsigned int *row);
```

To create new entities within the tree, *CBFlib* provides the functions:

```
int cbf_new_datablock (cbf_handle handle,
     const char *datablockname);
int cbf_new_category (cbf_handle handle,
     const char *categoryname);
int cbf_new_column (cbf_handle handle,
     const char *columnname);
int cbf_new_row (cbf_handle handle);
int cbf_insert_row (cbf_handle handle,
     unsigned int row);
int cbf_force_new_datablock (cbf_handle handle,
     const char *datablockname);
int cbf_force_new_category (cbf_handle handle,
     const char *categoryname);
```

The '`new`' functions add a new data block within the data set, a new category in the current data block, or a new column or row within the current category, and make it the current data block, category, column or row, respectively. If the data block, category or column already exists, then the function simply makes it the current data block, category or column. The function `cbf_new_row` adds the row to the end of the current category. `cbf_insert_row` provides the ability to insert a row before ordinal *row*, starting from 0. The newly inserted row gets the row ordinal *row* and the row that originally had that ordinal and all rows with higher ordinals are pushed downwards.

In general, CIF does not permit duplication of the names of data blocks or categories. In practice, however, duplications do occur. *CBFlib* provides '`force`' variants to allow creation of duplicate data-block and category names. Because, in this case, the program analysing the resulting file can only distinguish the duplicates by ordinal, these variants are not recommended for general use.

The following functions are used to remove entities from the tree:

```
int cbf_remove_datablock (cbf_handle handle);
int cbf_remove_category (cbf_handle handle);
int cbf_remove_column (cbf_handle handle);
int cbf_remove_row (cbf_handle handle);
int cbf_delete_row (cbf_handle handle,
     unsigned int row);
int cbf_reset_datablocks (cbf_handle handle);
int cbf_reset_datablock (cbf_handle handle);
int cbf_reset_category (cbf_handle handle);
```

The basic '`remove`' functions delete the current data block, category, column or row. Note that removing a data block makes the current data block, category, column and row undefined; removing a category makes the current category, column and row undefined. Removing a column makes the current column undefined, but leaves the current row intact, and removing a row leaves the current column intact. The function `cbf_delete_row` is similar to `cbf_remove_row` except that it removes the specified row in the current category. If the current row is not the deleted row, then it will remain valid.

All the categories in all data blocks, all the categories in the current data block or all the entries in the current category may be removed using the '`reset`' functions.

When a column and row within a category have been selected, the entry value may be examined or modified:

```
int cbf_get_value (cbf_handle handle,
     const char **value);
int cbf_set_value (cbf_handle handle,
     const char *value);
int cbf_get_integervalue (cbf_handle handle,
     int *number);
int cbf_set_integervalue (cbf_handle handle,
     int number);
int cbf_get_doublevalue (cbf_handle handle,
     double *number);
int cbf_set_doublevalue (cbf_handle handle,
     const char *format;
int cbf_get_integerarrayparameters (cbf_handle handle,
     unsigned int *compression, size_t *elsize,
     size_t *elements, int *maxelement);
int cbf_get_integerarray (cbf_handle handle,
     int *binary_id, int elsigned,
     size_t *elements_read);
int cbf_set_integerarray (cbf_handle handle,
     unsigned int compression, void *array,
     size_t elements);
```

A value within a CBF/imgCIF data set may be a simple character string, an integer or real number, or an array of integers. The functions `cbf_get_value` and `cbf_set_value` provide the basic functionality for normal CIF values, retrieving and modifying the

Table 5.6.2.5. *Values for the parameter* `compression` *in* `cbf_get_integerarrayparameters` *and* `cbf_set_integerarray`

`CBF_CANONICAL`	Canonical-code compression (Section 5.6.3.1)
`CBF_PACKED`	*CCP4*-style packing (Section 5.6.3.2)
`CBF_NONE`	No compression

current entry as a string. The functions `cbf_get_integervalue` and `cbf_get_doublevalue` interpret the retrieved string as an integer or real value and the functions `cbf_set_integer` and `cbf_set_doublevalue` convert the *number* argument into a string before setting the entry.

The functions for working with binary sections are more complicated as they must take into account compression, array size and the variety of different integer types available on different systems: signed/unsigned and various sizes.

The function `cbf_get_integerarrayparameters` retrieves the parameters of the current, binary, entry. The *compression* argument is set to the compression type used (Table 5.6.2.5). At present, this may take one of three values: `CBF_CANONICAL`, for canonical-code compression (see Section 5.6.3.1 below); `CBF_PACKED`, for *CCP4*-style packing (see Section 5.6.3.2 below); or `CBF_NONE`, for no compression. [*Note*: `CBF_NONE` is by far the slowest scheme of the three and uses much more disk space. It is intended for routine use with small arrays only. With large arrays (like images) it should be used only for debugging.] The *binary_id* value is a unique integer identifier for each binary section, *elsize* is the size in bytes of the array entries, *elsigned* and *elunsigned* are nonzero if the array can be read as unsigned or signed, respectively, *elements* is the number of entries in the array, and *minelement* and *maxelement* are the lowest and highest elements. If a destination argument is too small to hold a value, it will be set to the nearest value and the function will return `CBF_OVERFLOW`. If the current entry is not binary, `cbf_get_integerarrayparameters` will return `CBF_ASCII`.

`cbf_get_integerarray` reads the current binary entry into an integer array. The parameter *array* points to an array of elements interpreted as integers. Each element in the array is signed if *elsigned* is nonzero and unsigned otherwise, and each element occupies *elsize* bytes. The argument *elements_read* is set to the number of elements actually obtained. If the binary section does not contain sufficient entries to fill the array, the function returns `CBF_ENDOFDATA`. As before, the function will return `CBF_OVERFLOW` on overflow and `CBF_ASCII` if the entry is not binary.

`cbf_set_integerarray` sets the current binary or ASCII entry to the binary value of an integer array. As before, the acceptable values for compression are `CBF_PACKED`, `CBF_CANONICAL` and `CBF_NONE`. Each binary section should be given a unique integer identifier *binary_id*.

Two macros are provided to facilitate processing and propagation of error returns: one to return from the current function immediately and one to execute a given command first:

```
#define cbf_failnez(f) \
     {int err; err = (f); if (err) return err; }
#define cbf_onfailnez(f,c) \
     {int err; err = (f); if (err) {{c; }return err; }}
```

If the symbol `CBFDEBUG` is defined, alternative definitions that print out the error number as given in Table 5.6.1.1 are used:

```
#define cbf_failnez(x) \
{int err; err = (x); \
  if (err) { fprintf (stderr, \
     "\nCBFlib error %d in \"x\"\n", \
     err); return err; }}

#define cbf_onfailnez(x,c) \
{int err; err = (x); \
  if (err) { fprintf (stderr, \
    "\nCBFlib error %d in \"x\"\n", \
     err); \
     { c; } return err; }}
```

5.6.2.2. High-level *CBFlib* functions

The high-level *CBFlib* functions provide a level of abstraction above the CIF file structure and their prototypes are defined in the header file cbf_simple.h. Most of these functions simply use the low-level routines to navigate the CBF/imgCIF structure and read and modify data entries, and consequently expect a `cbf_handle` argument. There are also, however, additional sets of functions used to analyse the geometry of the goniometer and detector. These functions use additional handles of type `cbf_goniometer` and `cbf_detector`, respectively. All functions return the same error codes as the low-level functions do. The function return values are given in Table 5.6.1.1. The formal parameters for the high-level *CBFlib* functions are given in Table 5.6.2.6.

5.6.2.3. General high-level functions

The general high-level functions use the low-level routines to accomplish common tasks with a single call.

The first of these is used to facilitate the preparation of the complex CBF/imgCIF header structure:

```
int cbf_read_template (cbf_handle handle, FILE *file);
```

`cbf_read_template` simply reads the CBF/imgCIF file *file* into the data structure associated with the given handle and selects the first data block. It is typically used to read a template – an imgCIF file populated with data entries, but without any binary sections, into which experimental information can then be inserted. Template files are discussed further in Section 5.6.4 below.

The value of **`_diffrn_radiation_wavelength.wavelength`** can be retrieved or set. The functions

```
int cbf_get_wavelength (cbf_handle handle,
     double *wavelength);
int cbf_set_wavelength (cbf_handle handle,
     double wavelength);
```

operate on the categories DIFFRN_RADIATION and DIFFRN_RADIATION_WAVELENGTH. The wavelength is found indirectly. The value of **`_diffrn_radiation.wavelength_id`** is retrieved and used to find a matching row in the DIFFRN_RADIATION_WAVELENGTH category, from which the value of **`_diffrn_radiation_wavelength.wavelength`** is obtained.

The value of the ratio of the intensities of the polarization components **`_diffrn_radiation.polarizn_source_ratio`** and the value of the angle **`_diffrn_radiation.polarizn_source_norm`** between the normal to the polarization plane and the laboratory *Y* axis can be retrieved or set. The functions

```
int cbf_get_polarization (cbf_handle handle,
     double *polarizn_source_ratio,
     double *polarizn_source_norm);
int cbf_set_polarization (cbf_handle handle,
     double polarizn_source_ratio,
     double polarizn_source_norm);
```

operate on the DIFFRN_RADIATION category.

Table 5.6.2.6. *Formal parameters for high-level CBFlib functions*

`array`	Pointer to image array data
`axis_id`	Axis ID
`center1`	Displacement along the slow axis
`center2`	Displacement along the fast axis
`compression`	Compression type
`coordinate1`	*x* component
`coordinate2`	*y* component
`coordinate3`	*z* component
`crystal_id`	ASCII crystal ID
`day`	Timestamp day (1–31)
`detector`	Detector handle
`diffrn_id`	ASCII diffraction ID
`distance`	Distance
`div_x_source`	Value of **`_diffrn_radiation.div_x_source`**
`div_x_y_source`	Value of **`_diffrn_radiation.div_x_y_source`**
`div_y_source`	Value of **`_diffrn_radiation.div_y_source`**
`element_id`	ASCII element ID
`element_number`	Detector element counting from 0
`elements`	Count of elements
`elsigned`	Set to nonzero if the destination array elements are signed
`elsize`	Size in bytes of each destination array element
`file`	File descriptor
`gain`	Detector gain in counts per photon
`gain_esd`	Gain e.s.d. value
`goniometer`	Goniometer handle
`handle`	CBF handle
`hour`	Timestamp hour (0–23)
`increment`	Increment value
`index1`	Slow index
`index2`	Fast index
`initial1`	*x* component of the initial vector
`initial2`	*y* component of the initial vector
`initial3`	*z* component of the initial vector
`minute`	Timestamp minute (0–59)
`month`	Timestamp month (1–12)
`ndim1`	Slow array dimension
`ndim2`	Fast array dimension
`normal1`	*x* component of the normal vector
`normal2`	*y* component of the normal vector
`normal3`	*z* component of the normal vector
`overload`	Overload value
`polarizn_source_norm`	Polarization normal
`polarizn_source_ratio`	Polarization ratio
`precision`	Timestamp precision in seconds
`projected_area`	Apparent area in mm^2
`ratio`	Goniometer setting (0 = beginning of exposure, 1 = end)
`real1`	*x* component of the real-space vector
`real2`	*y* component of the real-space vector
`real3`	*z* component of the real-space vector
`reciprocal1`	*x* component of the reciprocal-space vector
`reciprocal2`	*y* component of the reciprocal-space vector
`reciprocal3`	*z* component of the reciprocal-space vector
`reserved`	Unused; any value other than 0 is invalid
`second`	Timestamp second (0–60.0)
`start`	Start value
`time`	Timestamp in seconds since 1 January 1970 or integration time in seconds
`timezone`	Time zone difference from universal time in minutes or `CBF_NOTIMEZONE`
`vector1`	*x* component of the rotation axis
`vector2`	*y* component of the rotation axis
`vector3`	*z* component of the rotation axis
`wavelength`	Wavelength in Å
`year`	Pointer to the destination timestamp year

The values of the divergence parameters, represented by the data names **`_diffrn_radiation.div_x_source`**, **`*.div_y_source`** and **`*.div_x_y_source`**, can be retrieved or set. The functions

```
int cbf_get_divergence (cbf_handle handle,
     double *div_x_source, double *div_y_source,
     double *div_x_y_source);
int cbf_set_divergence (cbf_handle handle,
     double div_x_source, double div_y_source,
     double div_x_y_source);
```

operate on the DIFFRN_RADIATION category.

The values of **`_diffrn.id`** and **`_diffrn.crystal_id`** can be retrieved or set:

```
int cbf_get_diffrn_id (cbf_handle handle,
     const char **diffrn_id);
int cbf_set_diffrn_id (cbf_handle handle,
     const char *diffrn_id);
int cbf_get_crystal_id (cbf_handle handle,
     const char **crystal_id);
int cbf_set_crystal_id (cbf_handle handle,
     const char *crystal_id);
```

Changing **`_diffrn.id`** also modifies the corresponding **`*.diffrn_id`** entries in the DIFFRN_SOURCE, DIFFRN_RADIATION, DIFFRN_DETECTOR and DIFFRN_MEASUREMENT categories.

The starting value and increment of an axis may be retrieved or set:

```
int cbf_get_axis_setting (cbf_handle handle,
     unsigned int reserved, const char *axis_id,
     double *start, double *increment);
int cbf_set_axis_setting (cbf_handle handle,
     unsigned int reserved, const char *axis_id,
     double start, double increment);
```

The `cbf_set_axis_setting` call is used during the creation of a CBF/imgCIF file to store the goniometer settings and rotation. The `cbf_get_axis_setting` is not generally useful when interpreting a file as there are no standard identifiers and the arrangement of the experimental axes is not consistent. Much more useful are the goniometer geometry functions described below.

The number of detector elements can be retrieved:

```
int cbf_count_elements (cbf_handle handle,
     unsigned int *elements);
```

This is the number of rows in the DIFFRN_DETECTOR_ELEMENT category. For each element, counting from 0, the detector identifier (the **`*.detector_id`** entry) can be retrieved and the gain and overload values in the ARRAY_INTENSITIES category retrieved or set:

```
int cbf_get_element_id (cbf_handle handle,
     unsigned int element_number,
     const char **element_id);
int cbf_get_gain (cbf_handle handle,
     unsigned int element_number,
     double *gain, double *gain_esd);
int cbf_set_gain (cbf_handle handle,
     unsigned int element_number, double gain,
     double gain_esd);
int cbf_get_overload (cbf_handle handle,
     unsigned int element_number, double *overload);
int cbf_set_overload (cbf_handle handle,
     unsigned int element_number, double overload);
```

For each element, counting from 0, the values of the parameters of the detector can be retrieved and some can be set. The value of **`_diffrn_detector_element.id`** is retrieved as *element_id*. The value of **`_diffrn_data_frame.array_id`** can be retrieved as *array_name*. The values of **`_array_intensities.gain`** and **`_array_intensities.gain_esd`** are retrieved as *gain*

and *gain_esd*. The value of **_array_intensities.overload** can be retrieved or set as *overload*. The value of **_diffrn_scan_frame.integration_time** can be retrieved or set as *integration_time*.

Timestamp calls operate on the DATE entry in the DIFFRN_SCAN_FRAME category:

```
int cbf_get_timestamp (cbf_handle handle,
     unsigned int reserved, double *time,
     int *timezone);
int cbf_set_timestamp (cbf_handle handle,
     unsigned int reserved, double time,
     int timezone, double precision);
int cbf_get_datestamp (cbf_handle handle,
     unsigned int reserved, int *year, int *month,
     int *day, int *hour, int *minute, double *second,
     int *timezone);
int cbf_set_datestamp (cbf_handle handle,
     unsigned int reserved, int year, int month,
     int day, int hour, int minute, double second,
     int timezone, double precision);
int cbf_set_current_timestamp (cbf_handle handle,
     unsigned int reserved, int timezone)
```

cbf_get_timestamp and cbf_set_timestamp measure time in seconds since 1 January 1970. cbf_get_datestamp and cbf_set_datestamp work in terms of individual *year*, *month*, *day*, *hour*, *minute* and *second*. The optional collection time zone, *timezone*, is the difference from universal time in minutes; *precision* is the fraction, in seconds, to which the time will be recorded. cbf_set_current_timestamp sets the collection timestamp from the current time, to the nearest second.

Also in the DIFFRN_SCAN_FRAME category is the integration time of the image:

```
int cbf_get_integration_time (cbf_handle handle,
     unsigned int reserved, double *time);
int cbf_set_integration_time (cbf_handle handle,
     unsigned int reserved, double time);
```

Finally, these functions include routines for working with binary images:

```
int cbf_get_image_size (cbf_handle handle,
     unsigned int reserved,
     unsigned int element_number,
     size_t *ndim1, size_t *ndim2);
int cbf_get_image (cbf_handle handle,
     unsigned int reserved,
     unsigned int element_number,
     void *array, size_t elsize, int elsign,
     size_t ndim1, size_t ndim2);
int cbf_set_image (cbf_handle handle,
     unsigned int reserved,
     unsigned int element_number,
     unsigned int compression, void *array,
     size_t elsize, int elsign, size_t ndim1,
     size_t ndim2);
```

cbf_get_image_size retrieves the dimensions of detector element *element_number* from the ARRAY_STRUCTURE_LIST category, setting *ndim1* and *ndim2* to the slow and fast array dimensions, respectively. These dimensions can be used to allocate memory before calling cbf_get_image. cbf_get_image reads the image data from detector element *element_number* into a signed or unsigned integer array of size *ndim1 * ndim2* and cbf_set_image associates image data with a detector element. As in the description of the integer array functions, the compression argument can currently take one of three values: CBF_CANONICAL, for canonical-code compression (see Section 5.6.3.1); CBF_PACKED, for *CCP4*-style packing (see Section 5.6.3.2); or CBF_NONE, for no compression.

5.6.2.4. Goniometer geometry functions

A CBF/imgCIF file includes a geometric description of the goniometer used to orient the sample during the experiment. Practical use of this information, however, is not trivial as it involves combining data from several categories and analysing in three dimensions the nested axes in which the description is framed (see Section 3.7.3 for a discussion of the axis system). *CBFlib* provides six functions to facilitate this task:

```
int cbf_construct_goniometer (cbf_handle handle,
     cbf_goniometer *goniometer);
int cbf_free_goniometer (cbf_goniometer goniometer);
int cbf_get_rotation_axis (cbf_goniometer goniometer,
     unsigned int reserved, double *vector1,
     double *vector2, double *vector3);
int cbf_get_rotation_range (cbf_goniometer goniometer,
     unsigned int reserved, double *start,
     double *increment);
int cbf_rotate_vector (cbf_goniometer goniometer,
     unsigned int reserved, double ratio,
     double initial1, double initial2, double initial3,
     double *final1, double *final2, double *final3);
int cbf_get_reciprocal (cbf_goniometer goniometer,
     unsigned int reserved, double ratio,
     double wavelength, double real1, double real2,
     double real3, double *reciprocal1,
     double *reciprocal2, double *reciprocal3);
```

cbf_construct_goniometer uses the data in the categories DIFFRN_MEASUREMENT, DIFFRN_MEASUREMENT_AXIS, AXIS, DIFFRN_SCAN_FRAME_AXIS and DIFFRN_SCAN_AXIS to construct a geometric representation of the goniometer and initializes the cbf_goniometer handle, *goniometer*. cbf_free_goniometer frees the goniometer structure. cbf_get_ rotation_axis and cbf_get_rotation_range get the normalized rotation vector, and the starting value and increment of the first rotating axis of the goniometer, respectively. The cbf_rotate_vector call applies the goniometer axis rotation to the given initial vector, with the *ratio* value specifying the goniometer setting from 0.0 at the beginning of the exposure to 1.0 at the end, irrespective of the actual rotation range. Finally, cbf_get_reciprocal transforms the given real-space vector (*real1, real2, real3*) to the corresponding reciprocal-space vector (*reciprocal1, reciprocal2, reciprocal3*). As before, the transform corresponds to the goniometer initial position with a *ratio* of 0.0 and the goniometer final position with a *ratio* of 1.0.

5.6.2.5. Detector geometry functions

In a similar manner, a CBF/imgCIF file includes a description of the surface of each detector and the arrangement of the pixels in space. *CBFlib* provides eight functions for analysing this description:

```
int cbf_construct_detector (cbf_handle handle,
     cbf_detector *detector,
     unsigned int element_number);
int cbf_free_detector (cbf_detector detector);
int cbf_get_beam_center (cbf_detector detector,
     double *index1, double *index2,
     double *center1, double *center2);
int cbf_get_detector_distance (cbf_detector detector,
     double *distance);
```

```
int cbf_get_detector_normal (cbf_detector detector,
     double *normal1, double *normal2,
     double *normal3);
int cbf_get_pixel_coordinates (cbf_detector detector,
     double index1, double index2,
     double *coordinate1, double *coordinate2,
     double *coordinate3);
int cbf_get_pixel_normal (cbf_detector detector,
     double index1, double index2,
     double *normal1, double *normal2,
     double *normal3);
int cbf_get_pixel_area (cbf_detector detector,
     double index1, double index2,
     double *area, double *projected_area);
```

`cbf_construct_detector` uses data from the categories DIFFRN, DIFFRN_DETECTOR, DIFFRN_DETECTOR_ELEMENT, DIFFRN_DETECTOR_AXIS, AXIS, ARRAY_STRUCTURE_LIST and ARRAY_STRUCTURE_LIST_AXIS to construct a geometric representation of detector element `element_number` and initializes the `cbf_detector` handle, `detector`. `cbf_free_detector` frees the detector structure; `cbf_get_beam_center` calculates the location at which the beam intersects the detector surface, either in terms of the pixel indices (`index1, index2`) along the slow and fast detector axes, respectively, or the displacement in millimetres along the slow and fast axes (`center1, center2`); `cbf_get_detector_distance` and `cbf_get_detector_normal` calculate the distance of the sample from the plane of the detector surface and the normal vector of the detector at pixel $(0, 0)$, respectively; `cbf_get_pixel_coordinates`, `cbf_get_pixel_normal` and `cbf_get_pixel_area` calculate the coordinates, normal vector, and area and apparent area as viewed from the sample position of the pixel with the given indices, respectively.

5.6.3. Compression schemes

Two schemes for lossless compression of integer arrays (such as images) have been implemented in this version of *CBFlib*:

(i) an entropy-encoding scheme using canonical coding;

(ii) a *CCP4*-style packing scheme.

Both encode the difference (or error) between the current element in the array and the prior element. Parameters required for more sophisticated predictors have been included in the compression functions and will be used in a future version of the library.

5.6.3.1. Canonical-code compression

The canonical-code compression scheme encodes errors in two ways: directly or indirectly. Errors are coded directly using a symbol corresponding to the error value. Errors are coded indirectly using a symbol for the number of bits in the (signed) error, followed by the error itself.

At the start of the compression, *CBFlib* constructs a table containing a set of symbols, one for each of the 2^n direct codes from -2^{n-1} to $2^{n-1} - 1$, one for a stop code and one for each of the $maxbits - n$ indirect codes, where n is chosen at compression time and *maxbits* is the maximum number of bits in an error. *CBFlib* then assigns to each symbol a bit code, using a shorter bit code for the more common symbols and a longer bit code for the less common symbols. The bit-code lengths are calculated using a Huffman-type algorithm and the actual bit codes are constructed using the canonical-code algorithm described by Moffat *et al.* (1997).

Table 5.6.3.1. *Structure of compressed data using the canonical-code scheme*

Byte	Value
1 to 8	Number of elements (64-bit little-endian number)
9 to 16	Minimum element
17 to 24	Maximum element
25 to 32	(Reserved for future use)
33	Number of bits directly coded, n
34	Maximum number of bits encoded, *maxbits*
35 to $35 + 2^n - 1$	Number of bits in each direct code
$35 + 2^n$	Number of bits in the stop code
$35 + 2^n + 1$ to $35 + 2^n + maxbits - n$	Number of bits in each indirect code
$35 + 2^n + maxbits - n + 1 \ldots$	Coded data

Table 5.6.3.2. *Structure of compressed data using the CCP4-style scheme*

Value in bits 3 to 5	Number of bits in each error
0	0
1	4
2	5
3	6
4	7
5	8
6	16
7	65

Byte	Value
1 to 8	Number of elements (64-bit little-endian number)
9 to 16	Minimum element (currently unused)
17 to 24	Maximum element (currently unused)
25 to 32	(Reserved for future use)
33 ...	Coded data

The structure of the compressed data is described in Table 5.6.3.1.

5.6.3.2. *CCP4*-style compression

The *CCP4*-style compression writes the errors in blocks. Each block begins with a 6-bit code. The number of errors in the block is 2^n, where n is the value in bits 0 to 2. Bits 3 to 5 encode the number of bits in each error. The data structure is summarized in Table 5.6.3.2.

5.6.4. Sample templates

The construction of CBF/imgCIF files can be simplified using templates. A template is itself an imgCIF file populated with data entries but without any binary sections. This file is normally associated with a CBF handle using the `cbf_read_template` call and provides a framework into which images and other experiment-specific data may be entered.

Fig. 5.6.4.1 is a sample template for an ADSC Quantum 4 detector (ADSC, 1997) with a κ-geometry diffractometer at Stanford Synchrotron Radiation Laboratory (SSRL) beamline 1-5.

The template for a MAR345 image plate detector (MAR Research, 1997) is almost identical. The major differences are in the size of the array (2300×2300 *versus* 2304×2304), the parameters for the CCD elements and the geometry of the elements. Therefore a few of the values in the AXIS, ARRAY_STRUCTURE_LIST, ARRAY_STRUCTURE_LIST_AXIS and ARRAY_INTENSITIES categories are different, as listed in Fig. 5.6.4.2.

```
###CBF: VERSION 1.1

data_image_1

# category DIFFRN
loop_
_diffrn.id
_diffrn.crystal_id
 DIFFRN_ID DIFFRN_CRYSTAL_ID

# category DIFFRN_SOURCE
loop_
_diffrn_source.diffrn_id
_diffrn_source.source
_diffrn_source.current
_diffrn_source.type
 DIFFRN_ID synchrotron 100.0 'SSRL beamline 1-5'

# category DIFFRN_DETECTOR_ELEMENT
loop_
_diffrn_detector_element.id
_diffrn_detector_element.detector_id
 ELEMENT1 ADSCQ4

# category DIFFRN_RADIATION
loop_
_diffrn_radiation.diffrn_id
_diffrn_radiation.wavelength_id
_diffrn_radiation.probe
_diffrn_radiation.monochromator
_diffrn_radiation.polarizn_source_ratio
_diffrn_radiation.polarizn_source_norm
_diffrn_radiation.div_x_source
_diffrn_radiation.div_y_source
_diffrn_radiation.div_x_y_source
_diffrn_radiation.collimation
 DIFFRN_ID WAVELENGTH1 x-ray 'Si 111' 0.8 0.0 0.08
    0.01 0.00 '0.20 mm x 0.20 mm'

# category DIFFRN_RADIATION_WAVELENGTH
loop_
_diffrn_radiation_wavelength.id
_diffrn_radiation_wavelength.wavelength
_diffrn_radiation_wavelength.wt
 WAVELENGTH1 0.98 1.0

# category DIFFRN_DETECTOR
loop_
_diffrn_detector.diffrn_id
_diffrn_detector.id
_diffrn_detector.type
_diffrn_detector.details
_diffrn_detector.number_of_axes
 DIFFRN_ID ADSCQ4 'ADSC QUANTUM4' 'slow mode' 4

 # category DIFFRN_DETECTOR_AXIS
loop_
_diffrn_detector_axis.detector_id
_diffrn_detector_axis.axis_id
 ADSCQ4 DETECTOR_X
 ADSCQ4 DETECTOR_Y
 ADSCQ4 DETECTOR_Z
 ADSCQ4 DETECTOR_PITCH

# category DIFFRN_DATA_FRAME
loop_
_diffrn_data_frame.id
_diffrn_data_frame.detector_element_id
_diffrn_data_frame.array_id
_diffrn_data_frame.binary_id
 FRAME1 ELEMENT1 ARRAY1 1
```

Fig. 5.6.4.1. Template imgCIF for use with an ADSC Quantum 4 detector.

```
# category DIFFRN_MEASUREMENT
loop_
_diffrn_measurement.diffrn_id
_diffrn_measurement.id
_diffrn_measurement.number_of_axes
_diffrn_measurement.method
_diffrn_measurement.details
 DIFFRN_ID GONIOMETER 3 rotation
; i0=1.000 i1=1.000 i2=1.000 ib=1.000 beamstop=20 mm
  0% attenuation
;


# category DIFFRN_MEASUREMENT_AXIS
loop_
_diffrn_measurement_axis.measurement_id
_diffrn_measurement_axis.axis_id
 GONIOMETER GONIOMETER_PHI
 GONIOMETER GONIOMETER_KAPPA
 GONIOMETER GONIOMETER_OMEGA


# category DIFFRN_SCAN
loop_
_diffrn_scan.id
_diffrn_scan.frame_id_start
_diffrn_scan.frame_id_end
_diffrn_scan.frames
 SCAN1 FRAME1 FRAME1 1


# category DIFFRN_SCAN_AXIS
loop_
_diffrn_scan_axis.scan_id
_diffrn_scan_axis.axis_id
_diffrn_scan_axis.angle_start
_diffrn_scan_axis.angle_range
_diffrn_scan_axis.angle_increment
_diffrn_scan_axis.displacement_start
_diffrn_scan_axis.displacement_range
_diffrn_scan_axis.displacement_increment
 SCAN1 GONIOMETER_OMEGA 0.0 0.0 0.0 0.0 0.0 0.0
 SCAN1 GONIOMETER_KAPPA 0.0 0.0 0.0 0.0 0.0 0.0
 SCAN1 GONIOMETER_PHI   0.0 0.0 0.0 0.0 0.0 0.0
 SCAN1 DETECTOR_Z       0.0 0.0 0.0 0.0 0.0 0.0
 SCAN1 DETECTOR_Y       0.0 0.0 0.0 0.0 0.0 0.0
 SCAN1 DETECTOR_X       0.0 0.0 0.0 0.0 0.0 0.0
 SCAN1 DETECTOR_PITCH   0.0 0.0 0.0 0.0 0.0 0.0

# category DIFFRN_SCAN_FRAME
loop_
_diffrn_scan_frame.frame_id
_diffrn_scan_frame.frame_number
_diffrn_scan_frame.integration_time
_diffrn_scan_frame.scan_id
_diffrn_scan_frame.date
 FRAME1 1 0.0 SCAN1 1997-12-04T10:23:48

# category DIFFRN_SCAN_FRAME_AXIS
loop_
_diffrn_scan_frame_axis.frame_id
_diffrn_scan_frame_axis.axis_id
_diffrn_scan_frame_axis.angle
_diffrn_scan_frame_axis.displacement
 FRAME1 GONIOMETER_OMEGA 0.0 0.0
 FRAME1 GONIOMETER_KAPPA 0.0 0.0
 FRAME1 GONIOMETER_PHI   0.0 0.0
 FRAME1 DETECTOR_Z       0.0 0.0
 FRAME1 DETECTOR_Y       0.0 0.0
 FRAME1 DETECTOR_X       0.0 0.0
 FRAME1 DETECTOR_PITCH   0.0 0.0
```

Fig. 5.6.4.1. (*cont.*)

```
# category AXIS
loop_
_axis.id
_axis.type
_axis.equipment
_axis.depends_on
_axis.vector[1] _axis.vector[2] _axis.vector[3]
_axis.offset[1] _axis.offset[2] _axis.offset[3]
 GONIOMETER_OMEGA rotation    goniometer
   . 1 0 0 . . .
 GONIOMETER_KAPPA rotation    goniometer
   GONIOMETER_OMEGA 0.64279 0 0.76604 . . .
 GONIOMETER_PHI   rotation    goniometer
   GONIOMETER_KAPPA 1 0 0 . . .
 SOURCE           general     source   . 0 0 1 . . .
 GRAVITY          general     gravity  . 0 -1 0 . . .
 DETECTOR_Z       translation detector
   . 0 0 -1 0 0 0
 DETECTOR_Y       translation detector
   DETECTOR_Z 0 1 0 0 0 0
 DETECTOR_X       translation detector
   DETECTOR_Y 1 0 0 0 0 0
 DETECTOR_PITCH   rotation    detector
   DETECTOR_X 0 1 0 0 0 0
 ELEMENT_X        translation detector
   DETECTOR_PITCH 1 0 0 -94.0032 94.0032 0
 ELEMENT_Y        translation detector
   ELEMENT_X 0 1 0 0 0 0

# category ARRAY_STRUCTURE_LIST
loop_
_array_structure_list.array_id
_array_structure_list.index
_array_structure_list.dimension
_array_structure_list.precedence
_array_structure_list.direction
_array_structure_list.axis_set_id
 ARRAY1 1 2304 1 increasing ELEMENT_X
 ARRAY1 2 2304 2 increasing ELEMENT_Y

# category ARRAY_STRUCTURE_LIST_AXIS
loop_
_array_structure_list_axis.axis_set_id
_array_structure_list_axis.axis_id
_array_structure_list_axis.displacement
_array_structure_list_axis.displacement_increment
 ELEMENT_X ELEMENT_X 0.0408 0.0816
 ELEMENT_Y ELEMENT_Y -0.0408 -0.0816

# category ARRAY_INTENSITIES
loop_
_array_intensities.array_id
_array_intensities.binary_id
_array_intensities.linearity
_array_intensities.gain
_array_intensities.gain_esd
_array_intensities.overload
_array_intensities.undefined_value
 ARRAY1 1 linear 0.23 0.03 65000 0


 # category ARRAY_STRUCTURE
loop_
_array_structure.id
_array_structure.encoding_type
_array_structure.compression_type
_array_structure.byte_order
 ARRAY1 "signed 32-bit integer" packed little_endian
```

Fig. 5.6.4.1. (*cont.*)

```
# category ARRAY_DATA
loop_
_array_data.array_id
_array_data.binary_id
_array_data.data
 ARRAY1 1 ?
```

Fig. 5.6.4.1. (*cont.*)

```
loop_
_axis.id
_axis.type
_axis.equipment
_axis.depends_on
_axis.vector[1] _axis.vector[2] _axis.vector[3]
_axis.offset[1] _axis.offset[2] _axis.offset[3]
 GONIOMETER_OMEGA rotation    goniometer
   . 1 0 0 . . .
 GONIOMETER_KAPPA rotation    goniometer
   GONIOMETER_OMEGA 0.64279 0 0.76604 . . .
 GONIOMETER_PHI   rotation    goniometer
   GONIOMETER_KAPPA 1 0 0 . . .
 SOURCE           general     source
   . 0 0 1 . . .
 GRAVITY          general     gravity
   . 0 -1 0 . . .
 DETECTOR_Z       translation detector
   . 0 0 -1 0 0 0
 DETECTOR_Y       translation detector
   DETECTOR_Z 0 1 0 0 0 0
 DETECTOR_X       translation detector
   DETECTOR_Y 1 0 0 0 0 0
 DETECTOR_PITCH   rotation    detector
   DETECTOR_X 0 1 0 0 0 0
 ELEMENT_X        translation detector
   DETECTOR_PITCH 1 0 0 -172.5 172.5 0
 ELEMENT_Y        translation detector
   ELEMENT_X 0 1 0 0 0 0

loop_
_array_structure_list.array_id
_array_structure_list.index
_array_structure_list.precedence
_array_structure_list.direction
_array_structure_list.axis_set_id
 ARRAY1 1 2300 1 increasing ELEMENT_X
 ARRAY1 2 2300 2 increasing ELEMENT_Y

 loop_
_array_structure_list_axis.axis_set_id
_array_structure_list_axis.axis_id
_array_structure_list_axis.displacement
_array_structure_list_axis.displacement_increment
 ELEMENT_X ELEMENT_X 0.075 0.150
 ELEMENT_Y ELEMENT_Y -0.075 -0.150

 loop_
_array_intensities.array_id
_array_intensities.binary_id
_array_intensities.linearity
_array_intensities.gain
_array_intensities.gain_esd
_array_intensities.overload
_array_intensities.undefined_value
 ARRAY1 1 linear 1.15 0.2 240000 0
```

Fig. 5.6.4.2. Part of the template file for a MAR345 detector. Values that differ from those in Fig. 5.6.4.1 are underlined.

```
   /* Read and modify the template */

cbf_failnez (cbf_make_handle (\&cbf))
in = fopen (template_name, "rb");

if (!in)
  exit (4);
cbf_failnez (cbf_read_template (cbf, in))

  /* Wavelength */

wavelength =
  img_get_number (img, "WAVELENGTH");
if (wavelength)
  cbf_failnez (
    cbf_set_wavelength (cbf, wavelength))

  /* Distance */

distance = img_get_number (img, "DISTANCE");
cbf_failnez (
  cbf_set_axis_setting (
    cbf, 0, "DETECTOR_Z", distance, 0))

  /* Oscillation start and range */

axis = img_get_field (img, "OSCILLATION AXIS");

if (!axis)

  axis = "PHI";

osc_start = img_get_number (img, axis);
osc_range = img_get_number (img,
   "OSCILLATION RANGE");
cbf_failnez (
  cbf_set_axis_setting (
    cbf, 0, "GONIOMETER_PHI",
      osc_start, osc_range))
```

Fig. 5.6.5.1. An extract of program code for the conversion of image data to CBF format.

5.6.5. Example programs

The *CBFlib* API comes with example programs and sample templates. The example program *convert_image* reads either of the templates described in Section 5.6.4, template_adscquantum4_2304x2304.cbf and template_mar345_2300x2300.cbf, and uses the higher-level *CBFlib* routines to convert an image file to a CBF.

The example programs *makecbf* and *img2cif* read an image file from a MAR300, MAR345 or ADSC CCD detector and then use *CBFlib* to convert that image to CBF format (*makecbf*) or to either imgCIF or CBF format (*img2cif*). *makecbf* writes the CBF-format image to disk, reads it in again, and then compares it with the original. *img2cif* just writes the desired file without a verification pass. *makecbf* works only from files on disk, so that random input/output can be used. *img2cif* includes code to process files from stdin and to stdout. The example program *cif2cbf* copies an ASCII CIF to a binary CBF/imgCIF file.

5.6.5.1. *convert_image*

The program *convert_image* takes two arguments, `imagefile` and `cbffile`. These are the primary input and output. The detector type is extracted from the image file, converted to lower case and used to construct the name of a template CBF to use for the copy. The template file name is of the form *template_name_columnsxrows*. The program comes with img.c and img.h to read image files. It uses the `img_...` routines to extract data and metadata from the image and uses the higher-level *CBFlib* routines to populate the template. Fig. 5.6.5.1 is a portion of the code that reads in the template and inserts the wavelength, the distance to the detector, and the oscillation start and range.

```
/* Make the _diffrn_frame_data category */
cbf_failnez(
  cbf_new_category         /* create the category */
    (cbf, "diffrn_frame_data"))
cbf_failnez(
  cbf_new_column           /* create the column   */
    (cbf, "id"))
cbf_failnez(
  cbf_set_value            /* add data            */
    (cbf, "frame_1"))
cbf_failnez(
  cbf_new_column           /* create next column  */
    (cbf, "detector_element_id"))
cbf_failnez(
  cbf_set_integervalue     /* add data            */
    (cbf, 1))
cbf_failnez(
  cbf_new_column           /* create the column   */
    (cbf, "detector_id"))
cbf_failnez(
  cbf_set_value            /* add data            */
    (cbf, detector_id))
cbf_failnez(
  cbf_new_column           /* create next column  */
   (cbf, "array_id"))
cbf_failnez(
  cbf_set_value            /* add data            */
    (cbf, "image_1"))
cbf_failnez(
  cbf_new_column           /* create next column  */
    (cbf, "binary_id"))
cbf_failnez(
  cbf_set_integervalue     /* add data            */
    (cbf, 1))
```

Fig. 5.6.5.2. Code fragment to illustrate the creation of the DIFFRN_FRAME_DATA category.

5.6.5.2. *makecbf*

makecbf is a good example of how to use many of the lower-level *CBFlib* functions. An example MAR345 image can be found at http://smb.slac.stanford.edu/~ellis/ and on the CD-ROM accompanying this volume. To run *makecbf* with the example image, type:

```
./bin/makecbf example.mar2300 test.cbf
```

The typical code fragment from makecbf.c shown in Fig. 5.6.5.2 creates the DIFFRN_FRAME_DATA category.

The program *img2cif* is an extended version of *makecbf* that allows the user to choose the details of the format of the output CBF. *img2cif* has the following command-line interface:

```
img2cif [-i input_image] [-o output_cif]
        [-c  {p[acked]|c[anonical]|[n[one]}]
        [-m  {h[eaders]|n[oheaders]}]
        [-d  {d[igest]|n[odigest]}]
        [-e {b[ase64]|q[uoted-printable]|
             d[ecimal]|h[exadecimal]|o[ctal]|n[one]}]
        [-b {f[orward]|b[ackwards]}]
        [input_image] [output_cif]
```

-i takes the name of the input image file in MAR300, MAR345 or ADSC CCD detector format. If no `input_image` file is specified or is given as '-', an image is copied from stdin to a temporary file. *-o* takes the name of the output CIF (if BASE64 or quoted-printable encoding is used) or CBF (if no encoding is used). If no `output_cif` is specified or is given as '-', the output is written to stdout. *-c* specifies the compression scheme (packed, canonical or none; the default is packed). *-m* selects MIME (Freed &

```
cbf_failnez (cbf_count_datablocks(cif, &blocks))
for (blocknum = 0; blocknum < blocks;  blocknum++)
{ /* start of copy loop */
  cbf_failnez (
    cbf_select_datablock(cif, blocknum))
  cbf_failnez (
    cbf_datablock_name(cif, &datablock_name))
  cbf_failnez (
    cbf_force_new_datablock(cbf, datablock_name))
  if ( !cbf_rewind_category(cif) ) {
    cbf_failnez (
      cbf_count_categories(cif, &categories))
    for (catnum = 0; catnum < categories; catnum++) {
      cbf_select_category(cif, catnum);
      cbf_category_name(cif,&category_name);
      cbf_force_new_category(cbf, category_name);
      cbf_count_rows(cif,&rows);
      cbf_count_columns(cif,&columns);
      /*  Transfer the column names from cif to cbf */
      if ( ! cbf_rewind_column(cif) ) {
      do {
        cbf_failnez(
          cbf_column_name(cif, &column_name))
        cbf_failnez(
          cbf_new_column(cbf, column_name))
      } while ( ! cbf_next_column(cif) );
      cbf_rewind_column(cif);
      cbf_rewind_row(cif);
      }
      /* Transfer the rows from cif to cbf */
      for (rownum = 0; rownum < rows; rownum++) {
        cbf_failnez (
          cbf_select_row(cif, rownum))
        cbf_failnez ( cbf_new_row(cbf))
        cbf_rewind_column(cif);
        for (colnum = 0; colnum < columns; colnum++) {
          cbf_failnez (cbf_select_column(cif, colnum))
          if ( ! cbf_get_value(cif, &value) ) {
            cbf_failnez (
              cbf_select_column(cbf, colnum))
            cbf_failnez ( cbf_set_value(cbf, value))
          } else {
            void * array;
            int binary_id, elsigned, elunsigned;
            size_t elements,elements_read, elsize;
            int minelement, maxelement;
            unsigned int cifcompression;

            cbf_failnez (
              cbf_get_integerarrayparameters(cif,
              &cifcompression, &binary_id, &elsize,
              &elsigned, &elunsigned, &elements,
              &minelement, &maxelement))
            if (array=malloc(elsize*elements)) {
              cbf_failnez (
                cbf_select_column(cbf,colnum))
              cbf_failnez (
                cbf_get_integerarray(
                cif, &binary_id, array, elsize,
                elsigned, elements, &elements_read))
              cbf_failnez(
                cbf_set_integerarray(
                cbf, compression, binary_id, array,
                elsize, elsigned, elements))
              free(array);
            } else {
              fprintf(stderr,
              "\nFailed to allocate memory %d bytes",
                elsize*elements);
               exit(1);
            }
          }
        }
      }
    }
  }
```

Fig. 5.6.5.3. Listing of the main code fragment in the program *cif2cbf* for conversion of an ASCII CIF to a CBF file.

Borenstein, 1996) headers within binary data value text fields. The default is headers for CIFs, no headers for CBFs. *-d* specifies the use of digests. The default is the MD5 digest (Rivest, 1992) when MIME headers are selected. *-e* specifies one of the standard MIME encodings: BASE64 (the default) or quoted-printable; or a non-standard decimal, hexadecimal or octal encoding for an ASCII CIF; or 'none' for a binary CBF. *-b* specifies the direction of mapping of bytes into words for decimal, hexadecimal or octal output, marked by '>' for forwards or '<' for backwards as the second character of each line of output, and in # comment lines. The default is backwards.

5.6.5.3. *cif2cbf*

The test program *cif2cbf* (Fig. 5.6.5.3) uses the same command-line options as *img2cif*, but accepts either a CIF or a CBF as input instead of an image file. The heart of the code is a series of nested loops. The outermost loop scans through all the data blocks. The next innermost loop scans through all the categories within each data block. The first of the two next innermost loops scans through the columns to copy the column headings. Then the second of these two loops scans through the rows. Finally, the innermost loop scans through the columns for each row.

We are grateful to Frances C. Bernstein for her helpful comments and suggestions.

References

ADSC (1997). *Quantum 4R specifications.* Poway, CA, USA: Area Detector Systems Corporation. http://www.adsc-xray.com/Q4techspecs.html.

Freed, N. & Borenstein, N. (1996). *Multipurpose Internet Mail Extensions (MIME) part one: format of internet message bodies.* RFC 2045. Internet Engineering Task Force. http://www.ietf.org/rfc/rfc2045.txt.

MAR Research (1997). *The mar345 image plate detector.* Norderstedt: MAR Research GmbH. http://www.marresearch.com/ip.htm.

Moffat, A., Bell, T. C. & Witten, I. H. (1997). *Lossless compression for text and images. Int. J. High Speed Electron. Syst.* **8**, 179–231.

Rivest, R. (1992). *The MD5 message-digest algorithm.* RFC 1321. Internet Engineering Task Force. http://www.ietf.org/rfc/rfc1321.txt.

Tosic, O. & Westbrook, J. D. (2000). *CIFParse. A library of access tools for mmCIF*. Reference guide. http://sw-tools.pdb.org/apps/CIFPARSE-OBJ/cifparse/index.html.

5.7. Small-molecule crystal structure publication using CIF

BY P. R. STRICKLAND, M. A. HOYLAND AND B. MCMAHON

5.7.1. Introduction

The International Union of Crystallography (IUCr) has always understood the importance of the accurate reporting of numerical results, and as far back as its early sponsorship of the Standard Crystallographic File Structure (Brown, 1983, 1988) the IUCr has explored the use of exchange files in publishing (see Chapter 1.1). In 1991, when the first draft of the CIF standard was nearing completion, the main journal of the IUCr for reporting crystal structures, *Acta Crystallographica Section C: Crystal Structure Communications* (hereafter *Acta Cryst. C*), consisted of a collection of concise reports of crystal and molecular structures presented in a standard format that would lend itself well to computerized markup and typesetting from an appropriate input file format. It seemed natural, therefore, to use this journal to test the new draft CIF standard and to develop techniques for machine-based checking of structural data along with the new methods for submitting, typesetting and distributing a crystal-structure report in electronic format. Although adopting a novel data-exchange format for the submission and handling of research papers might have seemed a radical and audacious development, the potential benefits in terms of accuracy and speed of publication were clear.

In parallel with the publication of the CIF standard (Hall *et al.*, 1991), an *Editorial* and revised *Notes for Authors* in *Acta Cryst. C* described the new route to publication using CIFs and invited the crystallographic community to cooperate in this innovative practice. The same issue of the journal contained the first paper to be published by this route (Willis *et al.*, 1991).

This first paper was the outcome of a testing phase which involved considerable interaction with the authors. The first unsolicited article to be submitted in CIF format appeared in the February 1992 issue of *Acta Cryst. C*. A few more were submitted during 1992, the number gradually increasing through the following year. Authors quickly adapted to the compartmentalized style of text entries and by the beginning of 1994 the level of CIF submissions allowed the journal to introduce a production stream that promised faster publication times for articles submitted electronically as CIFs. By the beginning of 1996, it became journal policy to accept *only* electronic submissions in CIF format.

The IUCr was not the only publisher to introduce the submission of structure reports in machine-readable form. In 1990, *Zeitschrift für Kristallographie*, published by R. Oldenbourg Verlag, introduced a new section for the publication of short inorganic and small-molecule structural papers with minimal commentary. To submit a report to this section, the author would use the output file from the refinement program *SHELX76* (Sheldrick, 1976) (at that time a *de facto* exchange standard on account of its widespread distribution), which was processed by a specially developed program *CASTOR* to create a self-contained file for use in publication. When CIF was introduced, it was also accepted as a submission format for this section of *Zeitschrift*. The section flourished and in 1997 it became a separate journal, *Zeitschrift für Kristallographie – New Crystal Structures*. CIF is now the standard submission format for this journal as well as for *Acta Cryst. C*.

In an era dominated by information retrieval *via* the world wide web, it is easy to forget that these innovations in crystallographic publishing predated the http protocol and the universal availability of graphical browsers. However, the independently developed but well defined CIF exchange standard proved easy to integrate with the publication procedures developed for electronic journals. The current delivery formats available to journals like *Acta Crystallographica* and *Zeitschrift für Kristallographie* are HTML and PDF. Nevertheless, the original CIF data are still accessible, and allow readers to visualize structures interactively in three dimensions or perform their own analyses of structural models.

The highly automated submission, checking and publication procedures of *Acta Cryst. C* and the online-only journal *Acta Crystallographica Section E: Structure Reports Online* (hereafter *Acta Cryst. E*) are described in detail in Section 5.7.2 as a case study for the publication of structure reports that are highly ordered in format. However, there are only a few journals that report detailed crystal structures and they represent a very specialized field of publishing. Section 5.7.3 discusses publications in which the reporting of structural data is only a minor or supplementary element of the article. It will become apparent that many of the considerations behind the design of a workflow for handling data-rich papers are also relevant to maximizing the value of data presented in or referenced by any scientific publication.

5.7.2. Case study: the fully automated reporting of small-unit-cell crystal structures

This section describes the route to publication of a small-molecule or inorganic single-crystal structure in *Acta Cryst. C* or *E* from the perspective of an author.

5.7.2.1. Assembling the complete article

For many authors the generation of a CIF suitable for publication is quite straightforward, since diffractometer software and structure solution and refinement packages have all been capable of writing or reading the CIF format for some time. In some highly integrated systems, the entire experimental, analysis and report-generating pathway may be controlled through a common user interface.

In other cases, different components must be collected from different sources and merged together, either by software utilities or, in the worst case, by hand-editing. It is a useful feature of the text-based CIF format that it can be modified by text editors or in certain word-processing modes; indeed, this was the only way in which the earliest CIF-based papers could be constructed. However, significant expertise and understanding of the technical details of the file format are needed to produce hand-edited files that are totally free from error. Authors are now encouraged to use software designed to help them create complete and error-free files (*e.g.* the *enCIFer* and *CIFEDIT* editors described in Chapter 5.3).

A complete structure communication comprises the following components.

Affiliations: PETER R. STRICKLAND, MICHAEL A. HOYLAND and BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

(*a*) Material common to the article as a whole:
  (i) title and authors;
  (ii) synopsis and/or abstract;
  (iii) comment section;
  (iv) acknowledgements;
  (v) references.

(*b*) Material relevant to each structure:
  (i) description of the experimental apparatus;
  (ii) description of the settings and environmental conditions for the experiment;
  (iii) experimental data, typically a list of measured and calculated structure factors for a single-crystal X-ray structure determination, or powder diffraction data with measured and calculated powder diffraction profiles;
  (iv) information about the compound, including source, preparation and formula;
  (v) summary of structure solution and refinement;
  (vi) coordinates of atomic sites, their elemental composition, occupancy, anisotropic displacement parameters, whether they are in part of the structure affected by positional disorder, and information about their refinement restraints;
  (vii) selected geometrical data.

(*c*) Graphical illustrations:
  (i) chemical structural diagrams;
  (ii) chemical diagrams of reaction pathways, tautomerism, bond properties *etc.*;
  (iii) crystallographic displacement-ellipsoid diagrams;
  (iv) crystallographic packing diagrams;
  (v) other graphs, plots or images.

Different journals will have different requirements for the arrangement of these items. For example, at the time of publication (2005), *Acta Crystallographica* requires that diffraction data (structure factors or Rietveld refinement profiles) are provided as supplementary information in separate files from that containing the body of the paper. This policy originated in the early days of network file transfer where relatively large files of experimental data could be transferred only with difficulty. This is less of a practical constraint now, and a case could be made for including the experimental results as an integral part of a single submission file, especially since there is still no formal mechanism in the core CIF dictionary to enforce an unambiguous connection between separate data blocks containing related data.

There is also not at present a standard way to include graphics within a CIF. The mechanisms of the imgCIF dictionary (Chapter 3.7) offer a possible approach to this problem. It is also possible to envisage the automated generation of views of the structure directly from the numerical data in the CIF. Three-dimensional ellipsoid plots are routinely generated from CIFs submitted to *Acta Crystallographica* for use in the review process and incomplete categories of data names exist in the core dictionary for the representation of two-dimensional diagrams of chemical connectivity. At present, however, neither of these is sufficiently well developed to generate publication-quality graphics in different orientations and styles as preferred by an author.

A journal may provide a *request list* of the data items that it considers recommended or mandatory. The request list for *Acta Cryst. C* and *E* is given in Appendix 5.7.1. An author can test a file intended for publication against a request list with a general-purpose CIF parsing tool such as *cif2cif* (Bernstein, 1998) or *QUASAR* (Hall & Sievers, 1993) (Chapter 5.3). Different request lists may be provided for different kinds of experiments, such as for powder diffraction experiments or for single-crystal studies using area detectors.

Note that an author always has the freedom to include additional data items in a CIF; the journal will exercise its own policy for the handling of data items not specified in its public request lists. The PUBL_MANUSCRIPT_INCL category available in the CIF core dictionary provides a mechanism for requesting the publication of data items that are not normally published by the journal (see Sections 5.7.2.3 and 3.2.5.5).

5.7.2.2. Reporting multiple structures and using templates

In CIF format, a data name cannot be repeated within a data block. Therefore, each structure reported in a CIF must occupy a separate data block. A journal might request a separate file for each structure; in the case of *Acta Cryst. C*, however, a single file for the entire submission is required. This file therefore contains several data blocks if the article reports several structures. The data-block codes (*i.e.* the changeable *label* part of a data-block header `data_label`) have no particular significance and are usually chosen by the authors as meaningful identifiers within their own collection of structures. However, each block code may be used *once only* in any individual file.

If an article reports only one structure, the author can include the general text of the article in the same data block that records the structure or in a separate data block. If the file already contains several data blocks (because it reports multiple structures), using a distinct data block for the text of the article is the most natural way of organizing the contents of the file. Fig. 5.7.2.1 shows the structure of a CIF that describes several structures.

Authors often have one or more local template data blocks that already include standard information about their contact details and details of the experiment. These templates may then be added or merged into the data blocks reporting the structures. Several standard crystallographic software packages include programs for merging CIF templates; one of the best known and most widespread is *SHELX97* (Sheldrick, 1997).

Some authors also use programmable macro facilities within commercial word-processing packages to achieve the same purpose. The IUCr application *printCIF for Word* (Westrip, 2004) extends this approach by creating a custom editing and formatting environment within Microsoft *Word*. These are very helpful utilities for authors who are not CIF experts. However, they are restricted to particular operating systems or software environments and are thus not universally available.

The program *enCIFer* (Allen *et al.*, 2004) provides facilities for importing templates and external files, and for adding and maintaining standard information about the authors of a CIF. It provides alternative representations of a CIF as a text file and as a collection of containers and object fields, and provides a great deal of support for authors who are not familiar with the technical details of the CIF format. *enCIFer* and other useful text-editing programs are described in Chapter 5.3.

5.7.2.3. Adding extra information to an article

An article for publication in *Acta Cryst. C* or *E* is built from a standard request list of CIF data items. Among the items included in this list are ones that describe molecular geometry: bond and contact distances, bond angles and torsion angles. In most cases, unexceptional values of these are not worth displaying (particularly as *Acta Cryst. C* and *E* make the original CIF data available as supplementary material). Authors can choose which values are to be displayed using a 'publication flag'. For example, the category

```
# ------ Material relevant to the whole paper ------
data_paper
_publ_section_title
; Two dimeric nickel(II) carboxylates
;
loop_
     _publ_author_name
     _publ_author_address
    'Morooka, Mitsuo'
; Department of Chemistry
  ...
;
    'Ohba, Shigeru'
; Department of Chemistry
  ...
;
_publ_section_abstract
; The structures of two dimeric Ni^II^ carboxylates
  and one dimeric Ni^II^ silanecarboxylate have been
  determined...
;

# ------Material relevant to the first structure ----
data_I
 _chemical_formula_sum            'C34 H54 N2 Ni2 O8'
 _chemical_formula_moiety         'C34 H54 N2 Ni2 O8'
 _chemical_formula_weight                      736.2
 _symmetry_cell_setting                   monoclinic
 loop_
     _symmetry_equiv_pos_as_xyz
 +x,+y,+z           -x,+y,1/2-z
 -x,-y,-z           +x,-y,1/2+z
 1/2+x,1/2+y,+z     1/2-x,1/2+y,1/2-z
 1/2-x,1/2-y,-z     1/2+x,1/2-y,1/2+z
 _symmetry_space_group_name_H-M           'C 1 2/c 1'
 _symmetry_space_group_name_Hall          '-C 2yc'
 _cell_length_a                             20.520(2)
 _cell_length_b                             10.647(1)
 _cell_length_c                             18.260(2)
 _cell_angle_alpha                             90.0
 _cell_angle_beta                           91.015(8)
 _cell_angle_gamma                             90.0

# --- Material relevant to the second structure -----
data_II
 _chemical_formula_sum        'C80 H68 C16 N2 Ni2 O8'
 _chemical_formula_moiety
                     'C78 H66 N2 Ni2 O8, 2(C H Cl3)'
 _chemical_formula_weight                     1515.5
 _symmetry_cell_setting                   triclinic
 loop_
     _symmetry_equiv_pos_as_xyz
 +x,+y,+z           -x,-y,-z
 _symmetry_space_group_name_H-M               'P -1'
 _symmetry_space_group_name_Hall              '-P 1'
 _cell_length_a                             13.231(1)
 _cell_length_b                             13.857(1)
 _cell_length_c                             11.425(1)
 _cell_angle_alpha                           99.48(1)
 _cell_angle_beta                           104.63(1)
 _cell_angle_gamma                          109.68(1)
```

Fig. 5.7.2.1. Structure of a CIF describing several crystal structures.

of data items that decribes bond lengths includes the data name `_geom_bond_publ_flag`, which may be assigned the value 'yes' or 'no' for any particular bond length depending on whether it should or should not be displayed.

The other items in the request list comprise the complete set of items that are by default extracted for publication from a CIF if they are present. An author may of course add more detail to an article within standard free-text fields (such as `_publ_section_comment`). However, if the additional information is present as a data item that is not in the standard request list, the typesetting software can be told to add this item dynamically to the request list, thus including the extra information in the published article. The way to do this is to list the additional data name or names as values of '`_publ_manuscript_incl_extra_item`'. The example below shows how to request that atom-site multiplicities and Wyckoff symbols are included in the table of atomic positions. These are data names defined in the core dictionary; this is indicated by the value '`yes`' of `_publ_manuscript_incl_extra_defn`.

```
loop_
  _publ_manuscript_incl_extra_item
  _publ_manuscript_incl_extra_defn
        '_atom_site_symmetry_multiplicity'    yes
        '_atom_site_Wyckoff_symbol'           yes
        '_Smith_crystal_magnetic_perm'         no
```

In this example, the author has also requested the publication of the value of the magnetic permeability of the crystal, which does not have a standard dictionary definition, but which has been recorded under a local data name, `_Smith_crystal_magnetic_perm`. Note that for this item, `_publ_manuscript_incl_extra_defn` takes the value '`no`'. The journal typesetting software has no procedure for handling arbitrary additional content, but it may be configured to recognize such a data name and typeset it in the desired style. Once the software is aware of this new item, it will automatically extract and format it in future submissions, as long as the author continues to list it under `_publ_manuscript_incl_extra_item`. It is best if the informal data name includes a registered reserved prefix (see Section 3.1.2.2), especially if machine-readable definitions are also provided in an appropriate DDL dictionary format and accessible through the IUCr register of CIF dictionaries (Section 3.1.8.2).

Care is needed when using `_publ_manuscript_incl_extra_item`:

(i) The extra items requested must be surrounded by quote marks, otherwise CIF software will try to interpret them as active data names.

(ii) The list is cumulative: if several `_publ_manuscript_incl_extra_item` loops appear in the file (one per data block), the request list that is generated will include all the extra items that appear in all of these loops, and that request list will be applied in full to all the data blocks in the file. It is therefore not possible to ask for an extra item from one data block but not another.

(iii) Not all possible terms in the official dictionaries may be recognized and handled appropriately by the journal software. To check this, the author can generate a preview of the formatted paper by using the *printcif* service, described in Section 5.7.2.4.

Two examples of this approach are shown in Fig. 5.7.2.2. Atom-site positions and displacement parameters are often displayed without the associated Wyckoff symbols or multiplicities (to save space). In the first example, the author indicates that the Wyckoff symbols should be displayed.

In the second example, the author wishes to publish a table of a set of items not defined in the core CIF dictionary (in this example, contact distances with associated charge density and Laplacian functions). Here, utility data names are used to extract regularly tabulated data of arbitrary content from the CIF to create a table in the published article.

5.7.2.4. Previewing the article

The appearance of the plain-text ordered arrangement of content in a CIF differs a great deal from its typeset representation in

```
loop_
_publ_manuscript_incl_extra_item
_publ_manuscript_incl_extra_defn
  '_atom_site_Wyckoff_symbol'          yes

loop_
  _atom_site_label
  _atom_site_type_symbol
  _atom_site_Wyckoff_symbol
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
Sn1 Sn 8g 0.20466(6) 0.28795(8)  0.2500     0.0104(2)
Ga1 Ga 8g 0.20466(6) 0.28795(8)  0.2500     0.0104(2)
Sn2 Sn 8f 0.0000     0.31416(11) 0.04613(8) 0.0120(3)
Ga2 Ga 8f 0.0000     0.31416(11) 0.04613(8) 0.0120(3)
Sm1 Sm 8e 0.20025(4) 0.0000      0.0000     0.0092(1)
Sn3 Sn 4c 0.0000     0.02592(13) 0.2500     0.0091(4)
Ga3 Ga 4c 0.0000     0.02592(13) 0.2500     0.0091(4)
Sm2 Sm 4c 0.0000     0.63989(8)  0.2500     0.0097(1)
```

(*a*)

```
loop_
_publ_manuscript_incl_extra_item
_publ_manuscript_incl_extra_defn
  '_geom_extra_tableA_col_1'           no
  '_geom_extra_tableA_col_2'           no
  '_geom_extra_tableA_col_3'           no
  '_geom_extra_tableA_col_4'           no
  '_geom_extra_table_head_A'           no
  '_geom_table_footnote_A'             no

_geom_extra_table_head_A
; Distances (l, \%A) and values for the charge
  density [\r(r), e \%A^-3^] and the Laplacian
  [\\nabla^2^\r(r), e/\%A^-5^] at the (3,-1) critical
  points for attractive shortened intramolecular
  contacts in (I).
;
loop_
_geom_extra_tableA_col_1
_geom_extra_tableA_col_2
_geom_extra_tableA_col_3
_geom_extra_tableA_col_4
Contact            l          \r(r)  \\nabla^2^\r(r)
N5...C8            2.813(5)   0.092       1.19
O6...N1            2.913(5)   0.079       0.95
N1...H16B          2.58       0.070       0.82
O1...H3            2.26       0.111       1.60
O5...H15           2.24       0.112       1.50
O7...H30B          2.34       0.095       1.28
O7...H31C          2.36       0.093       1.25
O7...H34B          2.36       0.092       1.24
O7...H35C          2.37       0.090       1.24

_geom_table_footnote_A        # maps to \tablefn
; The sums of the van der Waals radii (Zefirov &
  Zorky, 1989) are N and C = 3.21 \%A, N and
  O = 2.79 \%A, N and H = 2.66 \%A, and
  O and H = 2.45 \%A.
;
```

(*b*)

Fig. 5.7.2.2. Examples of authors' request-list extensions for items not normally printed in a paper. (*a*) Printing additional standard data items. The data are listed as normal in the ATOM_SITE loop. (*b*) A complete table of non-standard quantities associated with contact distances is generated, complete with table caption and footnote.

a journal article. It can help authors, therefore, if they can see how their article will appear in print (or as an online article) before they formally submit their article to a journal. *Acta Cryst. C* and *E* provide an online web service for this called *printcif* (http://journals.iucr.org/services/cif/printcif.html).

When an author uploads a CIF to the service, the data within it are extracted (using a dynamically enhanced request list if the publication of extra items has been requested) and translated through a sequence of software filters to TEX (Knuth, 1986). The TEX file is processed and a final document representation (a 'preprint') in PostScript or Portable Document Format (Adobe Systems Incorporated, 1999, 2004) is generated. The preprint is then downloaded to the author. The primary translation engine is the program *ciftex* (Section 5.3.5.3). However, *printcif* has additional content filters which are not distributed with *ciftex*; these are modified frequently to make additional pattern-based text substitutions or to make changes to the typographic style of the preprint to match any changes in the style of *Acta Cryst. C* or *E*.

A new approach to document formatting is being explored in the development of *printCIF for Word* (Westrip, 2004), an embedded Visual Basic application suitable for CIF editing and formatting within *Word* (Section 5.3.3.4.2). This allows users to preview their article as they work on it. However, *printCIF for Word* does not have access to the constantly updated translation filters used by *printcif*.

5.7.2.5. Data validation

The highly structured format of a CIF allows automated validation of the self-consistency and integrity of the structural data reported in it. What was traditionally a part of the referee's task in checking crystal structure papers can now be handled by software. *Acta Cryst. C* and *E* require authors to check their structures before submitting them for publication. The same checks are run on each CIF after submission and a report of the results is made available to the referees for use during the peer-review process.

The routine checking of submissions for errors was introduced by the IUCr journals in the early 1990s, initially as a manual procedure. When CIF was introduced, the new format was readily adopted as a standard interchange format from which the input files for different checking programs could be generated automatically. The development of a workflow based on CIF proved worthwhile, as CIF increasingly became the format for submission in the first place. Over time, too, much of the checking software became capable of reading CIFs directly, so that the intermediate data-conversion processes could be avoided.

Over several years, a great deal of experience was gained in the types of error that could most easily be detected using checking software. A major component of the checking suite was *UNIMOL*, which had been developed by the Cambridge Crystallographic Data Centre for checking the molecular geometry of database entries (Allen *et al.*, 1974). Other types of checks could be performed by running other general-purpose crystallographic packages under the direction of pre-defined scripts designed to exploit their particular strengths. Among the programs used in this way were *NRCVAX* (Gabe *et al.*, 1989), which incorporated the powerful *MISSYM* algorithm of Le Page (1988), *PARST* (Nardelli, 1983), an early version of *PLATON* (Spek, 1990) and the *BUNYIP* routine for detecting additional symmetry (Hester & Hall, 1996) within the *Xtal* program system (Hall *et al.*, 2000).

As experience grew in running these processes in increasingly automated ways, and in collecting, parsing and reformatting the most relevant diagnostic output, it became apparent that a modular system could be designed to perform most of the data checking entirely automatically. Preliminary work on the set of tests developed for the *PREPUB* component of the *Xtal* system (du Boulay & Hall, 1996) led, through close cooperation with the IUCr editorial office and Ton Spek, the author of *PLATON* (Spek, 2003), to the

implementation of *checkcif*, which is described in Section 5.7.2.6 below.

5.7.2.6. Automated data validation: *checkcif*

The current service for checking structural data submitted to IUCr journals is known as *checkcif* and is available at http://journals.iucr.org/services/cif/checkcif.html. Versions of this service have been made available to other publishers for some time. In 2003, a general service was introduced at http://checkcif.iucr.org to provide structural checks on CIF data sets destined for publication in non-IUCr journals or database deposition, or indeed to allow authors to assess the quality of their structure determinations whether they wish to publish them or not.

The tests carried out by *checkcif* include:

(i) a simple file syntax check: essential in the early days of manual CIF construction, but of less importance now as syntax-preserving editing programs have become more widespread;

(ii) tests for the self-consistency of mutually dependent data items present in the CIF;

(iii) a large collection of analytic tests on structural chemistry and molecular geometry based on the program *PLATON* (Spek, 2003).

The checks carried out at the time of publication (2005) are listed in Appendix 5.7.2 and on the CD-ROM accompanying this volume. The current list is available from http://journals.iucr.org/services/cif/datavalidation.html.

Although the results from *checkcif* provide valuable indications of possible inconsistencies or data errors, an article for publication is not accepted or rejected on the basis of the *checkcif* report alone. The report is always read by a reviewer as part of a considered critical appraisal of the article.

Sometimes, particular data values are so far from the expected values that some response is required from the author to explain them. The unusual values may be a consequence of poor experimental conditions that the author was unable to improve, or of poor crystal quality; they may indicate an uncertainty in part of the structure determination that the author considers acceptable, particularly if the purpose of the study is to concentrate on a different part of the structure; or they may genuinely indicate novel chemical features. Whatever the case, anomalous values usually need to be discussed by the author and the reviewer or editor, and often need to be commented on in the article. For *Acta Cryst. C* and *E*, *checkcif* generates in CIF format a list of the tests that have highlighted unusual values in the author's CIF (called 'A alerts'), together with a text field for each of these tests in which the author may justify or discuss the apparently anomalous results (see Fig. 5.7.2.3). Together these comprise a 'validation reply form'. The author can complete this form and paste it into the final version of the CIF submitted for publication. The editor handling the paper can then read the comments in the validation reply form and decide whether to accept the paper for publication. The submission system will automatically return to the author any CIF which generates an A alert but does not contain a completed validation reply form.

Every article published in *Acta Cryst. E* has as part of its supplementary material a summary of the *checkcif* report for the structure described in it. This summary includes any validation reply that the author has supplied. It also includes selected numerical data items identified by the journal editors as characterizing the overall quality and completeness of the structure determination.

The characterization of the 'quality' of a structure is a contentious issue. For journals, where there is active selection of articles for publication, it can be difficult to assign criteria for assessing the quality of the structure determination without these being seen as judging the quality or worth of the scientific work giving rise to the result. Thus journals rely upon the experience and discernment of referees to identify structures 'worth' publish-

```
The following ALERTS were generated. Each ALERT has the format
       test-name_ALERT_alert-type_alert-level.
Click on the hyperlinks for more details of the test.
Alert level A
PLAT035_ALERT_1_A No _chemical_absolute_configuration info given .          ?
PLAT761_ALERT_1_A CIF Contains no X-H Bonds ......................          ?
PLAT762_ALERT_1_A CIF Contains no X-Y-H or H-Y-H Angles ..........          ?
Alert level C
PLAT125_ALERT_4_C No _symmetry_space_group_name_Hall Given .......          ?
PLAT152_ALERT_1_C Supplied and Calc Volume s.u. Inconsistent .....          ?
PLAT707_ALERT_1_C D...A   Calc    3.324(2), Rep    3.321(2), Dev..       1.50 Sigma
            C13  -O4      1.555   2.556
Alert level G
REFLT03_ALERT_4_G Please check that the estimate of the number of Friedel pairs is
           correct. If it is not, please give the correct count in the
     _publ_section_exptl_refinement section of the submitted CIF.
          From the CIF: _diffrn_reflns_theta_max           27.00
          From the CIF: _reflns_number_total               2393
          Count of symmetry unique reflns         1395
          Completeness (_total/calc)            171.54          TEST3: Check
Friedels for noncentro structure
          Estimate of Friedel pairs measured       998
          Fraction of Friedel pairs measured     0.715
          Are heavy atom types Z>Si present        yes
   3 ALERT level A = In general: serious problem
   0 ALERT level B = Potentially serious problem
   3 ALERT level C = Check and explain
   1 ALERT level G = General alerts; check

   5 ALERT type 1 CIF construction/syntax error, inconsistent or missing data
   0 ALERT type 2 Indicator that the structure model may be wrong or deficient
   0 ALERT type 3 Indicator that the structure quality may be low
   2 ALERT type 4 Improvement, methodology, query or suggestion
```

(*a*)

Publication of your CIF

You should always attempt to resolve as many as possible of the alerts in all categories. Often the minor alerts point to easily fixed oversights, errors and omissions in your CIF or refinement strategy, so attention to these fine details can be worthwhile. In order to resolve some of the more serious problems it may be necessary to carry out additional measurements or structure refinements. However, the nature of your study may justify the reported deviations from the submission requirements of the journal and these should be commented upon in the discussion or experimental section of a paper - after all, they might represent an interesting feature.

If level A alerts remain, which you believe to be justified deviations, and you intend to submit this CIF for publication in Acta Crystallographica Section C or Section E, you should additionally insert an explanation in your CIF using the Validation Reply Form (VRF) below. Your explanation will be considered as part of the review process.

If you intend to submit to another section of Acta Crystallographica or Journal of Applied Crystallography or Journal of Synchrotron Radiation, you should make sure that at least a basic structural check is run on the final version of your CIF prior to submission.

```
# start Validation Reply Form
_vrf_PLAT035_99107abs
;
PROBLEM: No _chemical_absolute_configuration info given .          ?
RESPONSE: ...
;
_vrf_PLAT761_99107abs
;
PROBLEM: CIF Contains no X-H Bonds ......................          ?
RESPONSE: ...
;
_vrf_PLAT762_99107abs
;
PROBLEM: CIF Contains no X-Y-H or H-Y-H Angles ..........          ?
RESPONSE: ...
;
# end Validation Reply Form
```

If you wish to submit your CIF for publication in Acta Crystallographica Section C or E, you should upload your CIF via the web. If your CIF is to form part of a submission to another IUCr journal, you will be asked, either during electronic submission or by the Co-editor handling your paper, to upload your CIF via our web site.

(*b*)

Fig. 5.7.2.3. Extracts from a *checkcif* report for a 'publication check' on a CIF to be submitted to an IUCr journal. (*a*) Alerts of various levels of severity are listed. (*b*) The journal policy on the handling of alerts is summarized and a validation reply form listing the A alerts is supplied for the author to fill in.

ing. However, in a comprehensive collection of structural data sets, such as in a public structural database, it might be possible to identify particular data items that could be used for weighting individual data sets when the database is being 'mined' for particular patterns or characteristic values. It will be interesting to see whether a consensus emerges on what items would be suitable. It is clear that reliance on a single indicator will not be appropriate for sophisticated studies. The old idea that a structure could be classed as 'good' or 'bad' on the basis of its final residual *R* factor alone has long been abandoned, but it may be possible to stipulate criteria for a set of interrelated data items and use these to filter specific information from a database.

5.7.2.7. Submission and review

When an author has previewed and checked the contents of the CIF and has made the changes suggested by a careful study of the preprint and the *checkcif* report, the article may finally be submitted to *Acta Cryst. C* or *E* by file upload over the web. Other files completing or supporting the submission are also transferred to the editorial office at this time. These include structure-factor or powder profile listings for each structure, figures and chemical diagrams, and sometimes other supplementary documents. Structure-factor listings are supplied in CIF format. Figures may be in one of a number of standard graphics file formats, and at the moment have to be uploaded as separate files. Future extensions to CIF, perhaps following the imgCIF approach, may allow all the items needed to submit an article, including figures, to be prepared as a single file.

When all the files have arrived at the editorial office, a review document is generated that can be sent to the referees. This document contains: the text and tables of the article that will appear in the final publication, but laid out in a more open style suitable for annotation by hand; tables of atomic positions and geometry (containing all the data in the CIF, not just the subset that has been selected for displaying in the published article); certain fields from the CIF that are not normally printed but which may contain details of the way in which the experiment was carried out (these fields might have been completed manually or by the software controlling the experiment); the figures and other supplementary documents; and a print-out of the report from a final *checkcif* cycle, including a displacement-ellipsoid plot of the molecule in a minimal-overlap least-squares plane view. This composite document provides the information that a referee will typically want to consider in a compact and convenient form. Because the CIF is so highly structured, producing this review document is in most cases entirely automatic. The complete CIF as submitted by the author and the experimental data are also made available to the reviewer.

If revisions are requested, authors may upload modified files. The generation of revised versions of an article is also largely automatic.

5.7.2.8. Publication

When the final version of a CIF for *Acta Cryst. C* or *E* is approved, the article is ready for publication. Once more, the data fields required for the published article are extracted from the CIF and sorted. If the author has asked for additional items to be printed by using `_publ_manuscript_incl_extra_item`, these also are extracted. The result is transformed to a file suitable for processing by typesetting software. For *Acta Cryst. C* this was originally a TEX file; now a further transformation generates an SGML file that conforms to the document type definition (DTD) common to all IUCr journals. This allows not only typesetting and printing, but also the generation of the HTML for the navigable online version of the article, and the extraction of metadata for building online tables of contents and for supplying to bibliographic databases.

The conventional published article then appears in a monthly issue. Each article is still similar in style to the type of structure report published in journals for decades, although tables of atomic positions and geometric data are not usually displayed now, since these data are so readily available from the online article.

The online version of the journal, however, presents a much more information-rich version of the article. Each article is generally available in the form of a PDF file, suitable for downloading and offline printing. There is also an HTML version of the same text, and this version has rich internal links that make it easy to scroll back and forth through the article, jump to specific sections and see figures in low-resolution thumbnail or high-resolution views. The reference list contains links to the articles that are cited. There may also be links to related records in chemical or crystal structure databases. The reader may also download the experimental data and any supplementary documents associated with the article. As mentioned above, for *Acta Cryst. E* a summary of the check report is also available.

Finally, the structural data may be downloaded directly in CIF format. The CIF is presented in two ways. If a reader follows one link in a web browser, the file is interpreted simply as a text file and appears as a simple listing in the browser window, from which it may be printed or saved to disk. However, if the reader follows the other link, the CIF is transmitted to the browser with a header declaring its MIME type (Freed & Borenstein, 1996) as 'chemical/x-cif'. This is one of several MIME types registered for particular presentations of chemistry-related content by Rzepa & Murray-Rust (1998). The reader may then configure a web browser to respond in a specific way to content tagged with this MIME type; typically a helper application such as a molecular visualizer [*e.g. Mercury* (Bruno *et al.*, 2002)] will be launched that allows three-dimensional visualization and manipulation of the molecular or crystal structure.

When an article has been published in *Acta Cryst. C* or *E*, the CIF is transferred to the relevant public structural databases. Thus, the transcription errors that used to cause so many problems for data harvesters are completely avoided and one of the initial goals of the CIF project is achieved: uncorrupted data transfer from diffractometer, through publication, to a final repository.

Because *Acta Cryst. C* and *E* handle almost exclusively the publication of structure reports, the editorial workflow based on CIF lends itself to a very high level of automation and the journals are produced efficiently and on short timescales. Routine refereeing of structures is made very easy by the provision of checking reports, and the universal use of e-mail and web file transfer means that production times can be very fast.

5.7.3. CIF and other journals

Not every journal will be able to benefit to the same extent from the handling of CIFs. For many journals, structure reports will be secondary to the main purpose of most articles, and CIF data will more usually be deposited as supplementary or supporting documents, while only a summary (if anything) of the structure will be reported in the article body.

Nevertheless, the ability to extract data from CIFs automatically and the ability of much crystallographic software to read CIFs mean that even journals that do not specialize in crystallography can provide a production stream that includes careful checking of crystal structure data. The IUCr continues to develop *checkcif* as

a service which can be used by other publishers to enhance their checking of crystal structures, and there is considerable interest in this approach.

All journals publishing the results of crystal structure determinations may easily collect the supporting data in CIF format and transfer the files to public databases, improving the accuracy and efficiency of the database-building procedures.

5.7.3.1. Including CIF data in an article

For journals other than those specializing in full-scale structure reports, including CIF data in tables or reports of structures within general articles is rather more problematic. The translation of CIF data into XML seems to be a promising route to explore, as journals and reference volumes are increasingly being typeset from XML files. Traditionally, publishing has emphasized content markup that leads to a particular typographic representation. Modern trends are towards markup that tags the content by purpose, with the representation directed by external 'style files'. Consider Fig. 5.7.3.1, which shows the typeset representation of a set of data items in a CIF for a structural paper.

First, it can be seen that several CIF data items are omitted from the printed representation, such as the *International Tables* space-group number and the Hall symbol for the space group. For compactness, the printed data value does not have a legend or annotation if the meaning of an item is clear from the context; thus, the crystal system and Hermann–Mauguin space-group symbol are printed without any accompanying text. The journal may also omit information that is implicit given other data; thus the cell angles are not printed for an orthorhombic cell. On the other hand, units, which are implicit in the definition of a CIF data item, are printed. Related items are grouped together in a single expression, as in the case of the θ range or the crystal dimensions. In some cases, numerical values have been rounded to meet the journal's policy.

All of these transformations are matters of style, but it can be seen that they are not always trivial mappings to single data names. The style files determining the transformation from a detailed explicit data tabulation in the initial CIF may need to implement complex logical tests to suit the requirements of the journal.

Fig. 5.7.3.2 shows the same extract in TEX, the markup and typesetting language that was used for several years to produce *Acta Cryst. C*. It can be seen from this extract that the actual markup maps very closely to the initial CIF. All the cell parameters, including the cell angles, are present in the source file. The expansion of the macros (*e.g.* `\cellalpha`) executes the logic required to determine whether the value is to be printed and generates the additional text surrounding the value. Each data name is mapped to a distinct macro (even if the macros themselves have identical or near-identical internal structure), which preserves the semantic labelling of the original CIF. These macros are maintained in a separate file referenced and executed by every invocation of the typesetting program.

In contrast, Fig. 5.7.3.3 shows part of the SGML now used to typeset *Acta Cryst. C* and to generate HTML versions of the articles online. It is immediately seen that the markup emphasizes typographic style and positioning, and there is no explicit labelling by semantic element. Additional labelling is now found in the document structure; the individual items are marked up as 'list items' (`<li>`), but the arrangement of this list into a tabular form is a feature of the typesetting engine, not the SGML.

It is clear that the TEX macros provide a representation of the contents of the CIF that could easily be converted back to the initial input CIF. At present, such bidirectional translation is not possible from the SGML file.

Clearly, therefore, a mapping to SGML that preserved semantic markup would be preferable. It is most likely that suitable bidirectional translations would be based on XML.

```
data_I
_symmetry_cell_setting            orthorhombic
_symmetry_space_group_name_H-M    'P 21 21 21'
_symmetry_space_group_name_Hall   'P 2ac 2ab'
_symmetry_Int_Tables_number       19

loop_
 _symmetry_equiv_pos_as_xyz
 'x, y, z'
 '-x+1/2, -y, z+1/2'
 '-x, y+1/2, -z+1/2'
 'x+1/2, -y+1/2, -z'

_chemical_formula_moiety           'C10 H8 Br N S'
_diffrn_radiation_type              MoK\a
_exptl_absorpt_coefficient_mu       4.280

_cell_length_a                    5.7339(7)
_cell_length_b                    14.8229(15)
_cell_length_c                    23.469(2)
_cell_angle_alpha                 90.0
_cell_angle_beta                  90.0
_cell_angle_gamma                 90.0
_cell_volume                      1994.7(4)
_cell_formula_units_Z             8
_cell_measurement_temperature     296(2)
_cell_measurement_reflns_used     58
_cell_measurement_theta_min       4.806
_cell_measurement_theta_max       11.635

_exptl_crystal_description        plate
_exptl_crystal_colour             colourless
_exptl_crystal_size_max           0.40
_exptl_crystal_size_mid           0.32
_exptl_crystal_size_min           0.04
_exptl_crystal_density_meas       ?
_exptl_crystal_density_diffrn     1.693
_exptl_crystal_density_method     'not measured'
_exptl_crystal_F_000              1008
```

(*a*)

Crystal data

$C_{10}H_8BrNS$	Mo $K\alpha$ radiation
$M_r = 254.14$	Cell parameters from 58 reflections
Orthorhombic, $P2_12_12_1$	$\theta = 4.8$–11.6°
$a = 5.7339\ (7)$ Å	$\mu = 4.28$ mm^{-1}
$b = 14.8229\ (15)$ Å	$T = 296\ (2)$ K
$c = 23.469\ (2)$ Å	Plate, colourless
$V = 1994.7\ (4)$ Å^3	$0.40 \times 0.32 \times 0.04$ mm
$Z = 8$	
$D_x = 1.693$ Mg m^{-3}	

(*b*)

Fig. 5.7.3.1. Typesetting of structural data. The contents of the CIF (*a*) are transformed into a typeset representation (*b*) that omits, annotates or reorders the incoming data according to context and the style rules of the journal.

5.7.3.2. CIF and XML

XML is a specific concrete implementation of SGML suitable for generation of online browsable content. Mature style transformation mechanisms for XML exist and others are under active development.

```
\structno
\noindent{\nineit Crystal data}\nobreak\par
\vskip2pt\begindoublecolumns\twocoltrue\defaultfont
\raggedright
\everypar={\global\parindent=0pt\hangindent=1em
         \hangafter=1 }\noindent

\chemformiupac{C$_{10}$H$_{8}$BrNS}
\chemform{C$_{10}$H$_{8}$BrNS}
\chemformsum{C$_{10}$H$_{8}$BrNS}
\molwt{254.14}
\system{Orthorhombic}\def\sgsetno{2}
\defaultfont
\sgHM{$P2{}_{1}2{}_{1}2{}_{1}$}
\cella{5.7339 (7)}
\cellb{14.8229 (15)}
\cellc{23.469 (2)}
\cellalpha{90.00}
\cellbeta{90.00}
\cellgamma{90.00}
\cellvol{1994.7 (4)}
\cellz{8}
\dx{1.693}
\dm{missing}
\densmetha{not measured}
\radiationtype{Mo {\it K}$\alpha$}
\wavelength{0.71073}
\cellrefl{58}
\cellthetamin{4.8}
\cellthetamax{11.6}
\absorpmu{4.28}
\celltemp{296 (2)}
\shape{Plate}
\colour{Colourless}
\sizemax{0.40}
\sizemid{0.32}
\sizemin{0.04}
\sizerad{missing}
\origin{see text}
```

Fig. 5.7.3.2. Part of a TEX file used to print the article shown in Fig. 5.7.3.1(*b*).

```
<sec id="sec2.1">
 <sec id="sec2.1.1">
  <st><?print tpct=0pt>Crystal data</st>
  <p>
   <l id="l1" type="unord">
    <li><p>C<inf arrange="stagger">10</inf>H<inf
        arrange="stagger">8</inf>BrNS</p></li>
    <li><p><it>M</it><inf
        arrange="stagger"><it>r</it></inf> =
        254.14</p></li>
    <li><p>Orthorhombic, 
     <fi type="tex" print-info="tth" img.type="tth"
        img.data="teximages/ga1014fi3.tth">
        P2_{1}2_{1}2_{1}</fi></p></li>
    <li><p><it>a</it> =
        5.7339 (7) &Aring;</p></li>
    <li><p><it>b</it> =
        14.8229 (15) &Aring;</p></li>
    <li><p><it>c</it> =
        23.469 (2) &Aring;</p></li>
    <li><p><it>V</it> =
        1994.7 (4) &Aring;<sup
        arrange="stagger">3</sup></p></li>
    <li><p><it>Z</it> =
        8</p></li>
    <li><p><it>D</it><inf
        arrange="stagger"><it>x</it></inf> =
        1.693 Mg m<sup
        arrange="stagger">&minus;3</sup></p></li>
    <li><p>Mo <it>K</it>&alpha; radiation</p></li>
    <li><p>Cell parameters from 58 <?print show
        &softreturn;>reflections</p></li>
    <li><p>&thetas; = 4.8&ndash;11.6&deg;</p></li>
    <li><p>&mu; = 4.28 mm<sup
        arrange="stagger">&minus;1</sup></p></li>
    <li><p><it>T</it> =
         296 (2) K</p></li>
    <li><p>Plate, colourless</p></li>
    <li><p>0.40 &times; 0.32 &times;
         0.04 mm</p></li>
   </l>
 </p>
</sec>
```

Fig. 5.7.3.3. Part of the SGML file used to print the article shown in Fig. 5.7.3.1(*b*).

Section 5.3.8.2.1 describes one transformation to XML in the biological structures field, designed primarily for database interchange rather than publication. This transformation preserves the underlying data model of an mmCIF very closely, and one might anticipate similar XML transformations for small-molecule CIF applications and for publications. It is even possible that the XML transformations referred to in Chapter 5.3 could be used for publishing articles if suitable style transformations are developed, but this has not been tested yet.

One difficulty with a simple CIF-to-XML transformation is that it could be easily adapted to the publication of structure reports in dedicated journals, but would not necessarily be compatible with other XML implementations developed by an unspecialized publishing house. This could be avoided by the registration of an XML name space covering transformed CIF data and the production of portable stylesheet transformations that could be adopted and modified to meet the requirements of different publishing houses. As yet, we know of no initiatives in this direction.

XML name spaces have been registered to safeguard the development of subject-specific methods of representation as part of a project by the International Union of Pure and Applied Chemistry (Becker, 2001). One markup language that falls within the scope of this project is Chemical Markup Language (CML) (Murray-Rust & Rzepa, 1999, 2001).

Further discussions of the relationship between CIF and XML representations and a proposal for extensions to certain CIF data values to accommodate the wider range of data structures permitted in XML are given by Bernstein (2000).

We acknowledge the guidance, enthusiasm and dedication of past and present members of the editorial boards of *Acta Crystallographica Sections C* and *E* in developing the journals along the path described in this chapter. Particular tribute must be paid to Syd Hall, George Ferguson, Bill Clegg, David Watson and Tony Linden. We are very grateful to Ton Spek for his close involvement with the development of checking software, and also wish to acknowledge George Sheldrick, Mario Nardelli, Eric Gabe, Peter White, Yvon Le Page, Alan Mighell, Vicky Karen, Doug du Boulay, Mike Dacombe and Charlie Bugg for their help in the early days of automated structure checking. We wish also to pay tribute to the dedication and effort of our colleagues in the IUCr editorial office: Gillian Holmes, Sean Conway, Amanda Berry, Sarah Froggatt and Lisa Stephenson; and we thank the many authors who have been willing to test new approaches through the years.

Appendix 5.7.1
Request list for *Acta Crystallographica Section C*

Table A5.7.1.1 contains the request list for *Acta Crystallographica Section C* as given in the 2005 *Notes for Authors*. This list is appropriate for a single-crystal X-ray diffraction study and gives all the data items that are displayed in an article *if they are present in the CIF*. In principle, a smaller set of mandatory data items could be supplied as a separate request list. However, certain items may be

considered mandatory or not depending on the nature of the study and on the presence of other data items in the CIF, so checking for mandatory items is performed through higher-level algorithmic checks during the pre-submission validation stage.

Table A5.7.1.1. *Request list for Acta Crystallographica Section C*

(*a*) Data names relating to the text of an article	
`_publ_contact_author_name`	Contact author's name
`_publ_contact_author_address`	Contact author's address
`_publ_contact_author_email`	E-mail address to be published
`_publ_contact_author_fax`	For editorial communications
`_publ_contact_author_phone`	For editorial communications
`_publ_contact_letter`	Letter of submission, with date
`_publ_requested_journal`	'Acta Crystallographica Section C'
`_publ_requested_category`	Publication choice (FI, FM, FO, AD)
`_publ_section_title`	Title of paper
`_publ_section_title_footnote`	Footnote to title of paper
`_publ_author_name`	List of author(s) name(s)
`_publ_author_footnote`	Footnote(s) to author(s) name(s)
`_publ_author_address`	Author(s) address(es)
`_publ_section_synopsis`	Synopsis for compounds that cannot be shown as a chemical diagram
`_publ_section_abstract`	Abstract of paper in English
`_publ_section_comment`	Discussion of study
`_publ_section_acknowledgements`	Acknowledgements
`_publ_section_references`	References
`_publ_section_figure_captions`	Legends to figures
(*b*) Data names relating to the experimental data	
`_publ_section_exptl_prep`	Compound preparation details
`_chemical_formula_sum`	Chemical formula as sum of elements
`_chemical_formula_moiety`	Chemical formula in moieties
`_chemical_formula_weight`	Chemical formula mass (Da)
`_chemical_melting_point`	Melting point (K)
`_symmetry_cell_setting`	Code for cell setting
`_symmetry_space_group_name_H-M`	Space-group symbol, including unique axis
`_symmetry_equiv_pos_as_xyz`	Equivalent positions in order used by `_geom_`
`_cell_length_a`	Unit-cell lengths (Å)
`_cell_length_b`	
`_cell_length_c`	
`_cell_angle_alpha`	Unit-cell angles (°)
`_cell_angle_beta`	
`_cell_angle_gamma`	
`_cell_volume`	Unit-cell volume (Å^3)
`_cell_formula_units_Z`	Number of formulae per unit cell
`_exptl_crystal_density_diffrn`	Density calculated from unit cell and contents (Mg m^{-3})
`_exptl_crystal_density_meas`	Density measured experimentally (Mg m^{-3})
`_exptl_crystal_density_method`	Method used to measure density experimentally
`_diffrn_radiation_type`	Radiation type (*e.g.* neutron or Mo $K\alpha$)
`_diffrn_radiation_wavelength`	Radiation wavelength (Å)
`_cell_measurement_reflns_used`	Number of reflections used to measure unit cell
`_cell_measurement_theta_min`	Minimum θ of reflections used to measure unit cell (°)
`_cell_measurement_theta_max`	Maximum θ of reflections used to measure unit cell (°)
`_cell_measurement_temperature`	Measurement temperature (K)
`_exptl_absorpt_coefficient_mu`	Linear absorption coefficient (mm^{-1})
`_exptl_crystal_description`	Crystal habit description
`_exptl_crystal_size_max`	Maximum dimension of crystal (mm)
`_exptl_crystal_size_mid`	Medial dimension of crystal (mm)
`_exptl_crystal_size_min`	Minimum dimension of crystal (mm)
`_exptl_crystal_size_rad`	Radius of spherical or cylindrical crystal (mm)
`_exptl_crystal_colour`	Crystal colour
`_diffrn_measurement_device_type`	Diffractometer make and type
`_diffrn_measurement_method`	Mode of intensity measurement and scan
`_diffrn_detector_area_resol_mean`	Resolution of area detector (pixels mm^{-1})
`_exptl_absorpt_correction_type`	Code for absorption correction
`_exptl_absorpt_process_details`	Literature reference for absorption correction [*e.g.* '(North et al., 1968)']
`_exptl_absorpt_correction_T_min`	Minimum transmission factor from corrections
`_exptl_absorpt_correction_T_max`	Maximum transmission factor from corrections
`_diffrn_reflns_number`	Total number of reflections measured
`_reflns_number_total`	Number of symmetry-independent reflections
`_reflns_number_gt`	Number of reflections > σ threshold
`_reflns_threshold_expression`	σ expression for F, F^2 or I threshold
`_diffrn_reflns_theta_max`	Maximum θ of measured reflections (°)
`_diffrn_reflns_theta_full`	θ to which available reflections are 'complete' (°)
`_diffrn_measured_fraction_theta_max`	Fraction of unique reflections measured to θ_{max}
`_diffrn_measured_fraction_theta_full`	Fraction of unique reflections measured to θ_{full}
`_diffrn_reflns_av_R_equivalents`	R factor for symmetry-equivalent intensities
`_diffrn_reflns_limit_h_min`	Minimum/maximum h index of measured data
`_diffrn_reflns_limit_h_max`	
`_diffrn_reflns_limit_k_min`	Minimum/maximum k index of measured data
`_diffrn_reflns_limit_k_max`	
`_diffrn_reflns_limit_l_min`	Minimum/maximum l index of measured data
`_diffrn_reflns_limit_l_max`	
`_diffrn_standards_number`	Number of standards used in measurement
`_diffrn_standards_interval_count`	Number of measurements between standards
`_diffrn_standards_interval_time` *	Time (min) between standards
`_diffrn_standards_decay_%`	Percentage decrease in standards intensity
`_refine_ls_structure_factor_coef`	Code for F, F^2 or I used in least-squares refinement
`_refine_ls_R_factor_gt`	R factor of F for reflections > threshold
`_refine_ls_wR_factor_ref`	R factor of coefficient for refinement reflections
`_refine_ls_goodness_of_fit_ref`	Goodness of fit S for refinement reflections
`_refine_ls_number_reflns`	Number of reflections used in refinement
`_refine_ls_number_parameters`	Number of parameters refined
`_refine_ls_weighting_scheme`	Code for weight type
`_refine_ls_weighting_details`	Weighting expression
`_refine_ls_hydrogen_treatment`	Code for H-atom treatment
`_refine_ls_shift/su_max`	Maximum shift/s.u. ratio after final refinement cycle
`_refine_diff_density_max`	Maximum/minimum values of final difference map (e Å^{-3})
`_refine_diff_density_min`	
`_refine_ls_extinction_method`	Description of extinction methods applied
`_refine_ls_extinction_coef`	Extinction coefficient applied in corrections
`_refine_ls_abs_structure_details`	Absolute structure method and Friedel-pair number
`_refine_ls_abs_structure_Flack`	Measure of absolute structure
`_refine_ls_abs_structure_Rogers`†	Measure of absolute structure
`_publ_section_exptl_refinement`	Special details of the refinement
`_computing_data_collection`	Reference to data-collection software
`_computing_cell_refinement`	Reference to cell-refinement software
`_computing_data_reduction`	Reference to data-reduction software
`_computing_structure_solution`	Reference to structure-solution software
`_computing_structure_refinement`	Reference to structure-refinement software
`_computing_molecular_graphics`	Reference to visualization software
`_computing_publication_material`	Reference to publication preparation software
`loop_`	
`_atom_type_symbol`	Atom type symbol (usually element symbol)
`_atom_type_description`	Description of atom type
`_atom_type_scat_source`	Reference to scattering factors applied
`_atom_type_scat_dispersion_real`	Real anomalous-dispersion value applied
`_atom_type_scat_dispersion_imag`	Imaginary anomalous-dispersion value applied
`loop_`	
`_atom_site_label`	Unique label identifying the atom site
`_atom_site_fract_x`	Fractional coordinates of atom site
`_atom_site_fract_y`	
`_atom_site_fract_z`	
`_atom_site_U_iso_or_equiv`	Isotropic atomic displacement parameter, or equivalent from anisotropic atomic displacement parameters
`_atom_site_occupancy`	Occupancy fraction for site (default is 1.0)
`_atom_site_disorder_assembly`	Code that identifies functional group suffering disorder
`_atom_site_disorder_group`	Code that identifies disorder group
`_atom_site_adp_type`	Atomic displacement parameter type
`loop_`	
`_atom_site_aniso_label`	Unique label identifying the atom site
`_atom_site_aniso_U_11`	Elements of anisotropic atomic displacement parameter tensor
`_atom_site_aniso_U_22`	
`_atom_site_aniso_U_33`	
`_atom_site_aniso_U_12`	
`_atom_site_aniso_U_13`	
`_atom_site_aniso_U_23`	
`loop_`	
`_geom_bond_atom_site_label_1`	Labels identifying the atom sites 1 and 2
`_geom_bond_atom_site_label_2`	
`_geom_bond_site_symmetry_1`	Symmetry codes (*e.g.* 2_554) for atom sites 1 and 2
`_geom_bond_site_symmetry_2`	
`_geom_bond_distance`	Distance between atom sites 1 and 2 (Å)
`_geom_bond_publ_flag`	Flag for print request (yes or no)

* Alternative to `_diffrn_standards_interval_count`. † Alternative to `_refine_ls_abs_structure_Flack`.

Table A5.7.1.1 (*cont.*).

`loop_`	
`_geom_angle_atom_site_label_1`	Labels identifying the atom sites 1, 2 and 3
`_geom_angle_atom_site_label_2`	
`_geom_angle_atom_site_label_3`	
`_geom_angle_site_symmetry_1`	Symmetry codes for atom sites 1, 2 and 3
`_geom_angle_site_symmetry_2`	
`_geom_angle_site_symmetry_3`	
`_geom_angle`	Angle between atom sites 1, 2 and 3 (°)
`_geom_angle_publ_flag`	Flag for print request (yes or no)
`loop_`	
`_geom_torsion_atom_site_label_1`	Labels identifying the atom sites 1, 2, 3 and 4
`_geom_torsion_atom_site_label_2`	
`_geom_torsion_atom_site_label_3`	
`_geom_torsion_atom_site_label_4`	
`_geom_torsion_site_symmetry_1`	Symmetry codes for atom sites 1, 2, 3 and 4
`_geom_torsion_site_symmetry_2`	
`_geom_torsion_site_symmetry_3`	
`_geom_torsion_site_symmetry_4`	
`_geom_torsion`	Torsion angle between atom sites 1, 2, 3 and 4 (°)
`_geom_torsion_publ_flag`	Flag for print request (yes or no)
`loop_`	
`_geom_hbond_atom_site_label_D`	Donor-atom label in hydrogen bond
`_geom_hbond_atom_site_label_H`	H-atom label in hydrogen bond
`_geom_hbond_atom_site_label_A`	Acceptor-atom label in hydrogen bond
`_geom_hbond_site_symmetry_D`	Symmetry code for donor site
`_geom_hbond_site_symmetry_H`	Symmetry code for H-atom site
`_geom_hbond_site_symmetry_A`	Symmetry code for acceptor site
`_geom_hbond_distance_DH`	Donor atom-to-H-atom distance (Å)
`_geom_hbond_distance_HA`	H-atom-to-acceptor atom distance (Å)
`_geom_hbond_distance_DA`	Donor atom-to-acceptor atom distance (Å)
`_geom_hbond_angle_DHA`	Donor to H to acceptor angle (°)
`_geom_hbond_publ_flag`	Flag for print request (yes or no)
(*c*) Data names for adding items to the standard request list	
`loop_`	
`_publ_manuscript_incl_extra_item`	Additional CIF item submitted for publication
`_publ_manuscript_incl_extra_defn`	Is item defined in core dictionary? (yes or no)
(*d*) Data names for structure-factor lists	
`loop_`	
`_refln_index_h`	Miller indices *h*, *k* and *l*
`_refln_index_k`	
`_refln_index_l`	
`_refln_F_meas`	Measured F
`_refln_F_squared_meas`‡	Measured F^2
`_refln_F_sigma`	Standard uncertainty of F
`_refln_F_squared_sigma`‡	Standard uncertainty of F^2
`_refln_F_calc`	Calculated F
`_refln_F_squared_calc`‡	Calculated F^2

‡ Alternative to the corresponding data name without 'squared'.

Appendix 5.7.2
Data validation using *checkcif*

Table A5.7.2.1 lists the *checkcif* tests concerned primarily with the completeness and self-consistency of individual or closely related data items. These tests were developed from the routines of *PREPUB* (du Boulay & Hall, 1996) and in the IUCr Editorial Office. Table A5.7.2.2 lists the tests applied specifically by the program *PLATON* (Spek, 2003), which performs a more detailed crystallographic analysis of the structure itself.

Each entry in each table has an identifying code and a numeric type. The type is used to categorize the alert messages generated when the tested values deviate from assigned norms. Type 1 refers to syntactic or other errors of construction in the CIF, or to inconsistent or missing data. Type 2 alerts indicate that the structure model may be wrong or deficient. Type 3 alerts indicate that the quality of the structure may be low, owing to limited or incomplete data coverage. Alerts of type 4 are indicative of deviations from style or suggested good practice, or may offer suggestions for improvement in presentation. The alerts within each category may be of varying levels of severity.

Full details of the tests and algorithms applied for the *checkcif* tests may be found at http://journals.iucr.org/services/cif/datavalidation.html or on the CD-ROM accompanying this volume. These include comments which provide help in interpreting the results of the tests and suggest ways in which the author can improve the data. The comments were provided by A. Linden and other members of the IUCr journal editorial boards.

The tests listed in Tables A5.7.2.1 and A5.7.2.2 are appropriate for small-unit-cell single-crystal structure determinations. More discriminating tests are being introduced for powder diffraction studies and for modulated structures.

Table A5.7.2.1. *List of data-validation tests applied by checkcif*

Test name	Type	Purpose
ABSMU01	1	Check that μ is consistent with the cell contents
ABSTM01	1	Check that T_{min} is less than T_{max}
ABSTM02	3	Check that T_{min} and T_{max} are appropriate to the crystal size and μ
ABSTY01	1	Check that `_exptl_absorpt_correction_type` is a recognized keyword
ABSTY02	1	Check that `_exptl_absorpt_correction_type` contains some reference text
CELLK01	1	Check that temperature is in Kelvin
CELLT01	1	Check that θ_{min} is less than θ_{max}
CELLV01	1	Check that the `_cell_volume` matches `_cell_length_` and `_cell_angle_` values
CELLV02	1	Check that the `_cell_volume` s.u. matches `_cell_length_` and `_cell_angle_` s.u. values
CELLZ01	1	Check consistency between formula, *Z*, atom list and symmetry
CHEMS01	1	Check that the `_chemical_formula_sum` is properly constructed
CHEMS02	1	Check that the stated category is consistent with the formula of the compound
CHEMW01	1	Check consistency between `_chemical_formula_weight` and `_chemical_formula_sum`
CHEMW03	2	Check consistency between weight, *Z*, symmetry and atom list
CRYSC01	1	Check that colour of crystal is consistent with expected colour code combinations
CRYSR01	1	Check that the radius of the crystal is given for a spherical or cylindrical crystal
CRYSS01	1	Check consistency of crystal dimensions
CRYSS02	3	Check that the values of `_exptl_crystal_size_`* are not larger than expected
DENSD01	1	Check consistency of density, cell volume and weight
DENSX01	1	Check that `_exptl_crystal_density_meas` matches `_exptl_crystal_density_diffrn`
DIFMN01	1	Check that `_refine_diff_density_min` is less than `_refine_diff_density_max`
DIFMN02	2	Check that `_refine_diff_density_min` is within expected limits
DIFMN03	1	Check for adjacent site information if `_refine_diff_density_min` is outside expected limits
DIFMX01	2	Check that `_refine_diff_density_max` is within expected limits
DIFMX02	1	Check for adjacent site information if `_refine_diff_density_max` is outside expected limits
FCOEF01	1	Check that the value of `_refine_ls_structure_factor_coef` is recognized
FORMU01	1/2	Check consistency between formulae and atom site data
GOODF01	1/2	Check that `_refine_ls_goodness_of_fit_ref` is within expected limits
HYDTR01	1	Check that the value of `_refine_ls_hydrogen_treatment` is recognized
RADNT01	1	Check that the radiation type is recognized
RADNW01	1	Check that `_diffrn_radiation_wavelength` matches `_diffrn_radiation_type`
REFLE01	3	Check that `_reflns_threshold_expression` contains a multiplier which is below the limit
REFLG01	1	Check that `_reflns_number_gt` is less than or equal to `_diffrn_reflns_number`
REFLL01	1	Check that `_diffrn_reflns_limit_` values are in the correct order
REFLT01	1	Check that `_reflns_number_total` is less than or equal to `_diffrn_reflns_number`
REFLT02	1	Check that `_reflns_number_total` is greater than or equal to `_reflns_number_gt`
REFLT03	1/3/4	Check consistency of `_reflns_number_total` with cell volume, symmetry and θ_{max}
REFNR01	3	Check the ratio of `_refine_ls_number_reflns` and `_refine_ls_number_parameters`
RFACG01	3	Check that `_refine_ls_R_factor_gt` is within expected limits
RFACR01	3	Check that `_refine_ls_wR_factor_ref` is within expected limits
RINTA01	3	Check that `_diffrn_reflns_av_R_equivalents` is within expected limits
SHFSU01	2	Check that `_refine_ls_shift/su_max` is within expected limits
STRDE01	1	Check that `_refine_ls_abs_structure_details` is present if necessary
STRVA01	2/4	Check that `_refine_ls_abs_structure_Flack` is within expected limits
STRVA02	2/3/4	Check that `_refine_ls_abs_structure_Rogers` is within expected limits
SYMMG01	1	Check that the `_symmetry_space_group_name_H-M` value is recognized

Table A5.7.2.1 (*cont.*).

SYMMG02	1	Check consistency between space group name and symmetry positions
SYMMS01	1	Check that the **_symmetry_cell_setting** matches one of the keywords
SYMMS02	1	Check consistency between cell setting and cell parameters
THETM01	3	Check that **_diffrn_reflns_theta_max** is greater than expected limits
WEIGH01	1	Check that the value of **_refine_ls_weighting_scheme** is recognized

Table A5.7.2.2. *List of data-validation tests applied by PLATON*

Test name	Type	Purpose
PLAT020	3	Check *R*(int)
PLAT021	1	Check expected number of reflections (max = 1 centro, 2 non-centro)
PLAT022	3	Check expected number of reflections
PLAT023	3	Check $\theta_{\max}$
PLAT024	4	Check for required Friedel pair averaging $Z \leq$ Si
PLAT025	1	Check for $h_{\min} \ldots l_{\max}$
PLAT026	3	Check for weak data
PLAT027	3	Check **_diffrn_reflns_theta_full**
PLAT028	3	Check **_diffrn_measured_fraction_theta_max**
PLAT029	3	Check **_diffrn_measured_fraction_theta_full**
PLAT030	1	Check **_diffrn_reflns_number** > **_reflns_number_total**
PLAT031	4	Check need for extinction correction parameter
PLAT032	4	Check s.u. Flack parameter
PLAT033	2	Check Flack parameter value
PLAT034	1	Check for Flack parameter value specified $Z >$ Si, non-centro
PLAT035	1	Check for **_chemical_absolute_configuration**
PLAT036	1	Check for missing Flack parameter s.u.
PLAT037	1	Check **_diffrn_reflns_theta_full**
PLAT038	1	Check **_diffrn_measured_fraction_theta_max**
PLAT039	1	Check **_diffrn_measured_fraction_theta_full**
PLAT040	1	Test for H atoms [0, 1]
PLAT041	1	Test sum formula
PLAT042	1	Test moiety formula
PLAT043	1	Test for molecular weight
PLAT044	1	Check reported with calculated density
PLAT045	1	Check reported and calculated *Z*
PLAT046	1	Check reported density with calculated density from $Z * MW$
PLAT047	1	Test sum formula given
PLAT048	1	Test moiety formula given
PLAT049	1	Check calculated density > 1.0
PLAT050	1	Test for μ given [0, 1]
PLAT051	1	Test for difference μ(cif) with μ(calc) [%]
PLAT052	1	Test for specification of absorption correction method [0, 1]
PLAT053	1	Test for specification crystal dimension min [0, 1]
PLAT054	1	Test for specification crystal dimension mid [0, 1]
PLAT055	1	Test for specification crystal dimension max [0, 1]
PLAT056	1	Test for specification crystal radius [0, 1]
PLAT057	3	Test for correction for absorption needed
PLAT058	1	Test for specification $T_{\max}$ [0, 1]
PLAT059	1	Test for specification $T_{\min}$ [0, 1]
PLAT060	3	*RR* test
PLAT061	3	RR' test
PLAT062	4	Rescale $T_{\min}$ and $T_{\max}$
PLAT063	3	Test for crystal size
PLAT064	1	Test for $T_{\max} \geq T_{\min}$
PLAT065	3	Test for applicability of (semi-)empirical absorption correction [0, 1]
PLAT066	1	Test whether predicted and reported transmission ranges are identical
PLAT067	1	Ensure that minimum dimension < max dimension
PLAT068	1	Test for $F(000)$ calc/reported difference
PLAT070	1	Test for duplicate labels
PLAT071	1	Test for uninterpretable labels
PLAT074	1	Test for occupancy = 0.0
PLAT075	1	Test for occupancy greater than 1.0
PLAT076	1	Test for occupancy less than 1.0 for atom on special position
PLAT077	4	Test for non-integral number of atoms in unit cell
PLAT080	2	Test maximum shift/error
PLAT081	1	Test for maximum shift/error given
PLAT082	2	Test for reasonable *R*1
PLAT083	2	Test for extreme second weighting parameter (*SHELXL*)
PLAT084	2	Test for reasonable *wR*2
PLAT085	2	Test for default *SHELXL* weighting scheme
PLAT086	2	Test for reasonable *S* (too low)
PLAT087	2	Test for reasonable *S* (too high)
PLAT088	3	Test for reasonable data/parameter ratio (centro)
PLAT089	3	Test for reasonable data/parameter ratio (non-centro) ($Z_{\max} < 18$)
PLAT090	3	Test for reasonable data/parameter ratio (non-centro) ($Z_{\max} > 18$)
PLAT091	1	Test for 'No wavelength given'
PLAT092	4	Test for wavelength type [Cu, Mo, Ag] [0, 1]
PLAT093	1	Test for inconsistency 'mixed' *versus* 'no refined H'
PLAT094	2	Test for maximum/minimum residual density ratio
PLAT095	1	Test for residual density maximum given [0, 1]
PLAT096	1	Test for residual density minimum given [0, 1]
PLAT097	2	Test maximum residual density
PLAT098	2	Test for minimum residual density

Table A5.7.2.2 (*cont.*).

Test name	Type	Purpose
PLAT099	1	Test for minimum residual density less than zero [0, 1]
PLAT110	2	Test for additional translational symmetry [0, 1]
PLAT111	2	Test for additional centre of symmetry [0, 100]
PLAT112	2	Test for additional symmetry [0, 1]
PLAT113	2	Report new space group suggested by *ADDSYM*
PLAT114	2	Report on *ADDSYM* problem
PLAT120	1	Test for consistent **_symmetry_space_group_name_H-M** and symmetry operations
PLAT121	1	Test for valid **_symmetry_space_group_name_H-M**
PLAT122	1	Test for ? **_symmetry_space_group_name_H-M**
PLAT123	1	Test for interpretable space-group symmetry
PLAT124	1	Test for **_symmetry_equiv_pos_as_xyz** present
PLAT125	4	Test for ? **_symmetry_space_group_name_Hall**
PLAT126	1	Test for **_symmetry_space_group_name_Hall** error
PLAT127	1	Test for **_symmetry_space_group_name_Hall** consistency
PLAT128	4	Test for non-standard monoclinic space-group setting
PLAT129	4	Test for unusual non-standard space-group name
PLAT130	1	Test for cubic: $a = b = c$
PLAT131	1	Test for cubic: $\alpha = \beta = \gamma = 90$
PLAT132	1	Test for trigonal/hexagonal: $a = b$
PLAT133	1	Test for trigonal/hexagonal: $\alpha = \beta = 90$
PLAT134	1	Test for trigonal/hexagonal: $\gamma = 120$
PLAT135	1	Test for tetragonal: $a = b$
PLAT136	1	Test for tetragonal: $\alpha = \beta = \gamma = 90$
PLAT137	1	Test for orthorhombic: $\alpha = \beta = \gamma = 90$
PLAT138	1	Test for monoclinic more than 1 angle off 90 degrees
PLAT139	1	Test for rhombohedral $a = b = c$
PLAT140	1	Test for rhombohedral $\alpha = \beta = \gamma$
PLAT141	4	S.u. on *a* axis small or missing
PLAT142	4	S.u. on *b* axis small or missing
PLAT143	4	S.u. on *c* axis small or missing
PLAT144	4	S.u. on α small or missing
PLAT145	4	S.u. on β small or missing
PLAT146	4	S.u. on γ small or missing
PLAT147	1	S.u. on symmetry restricted cell angle
PLAT150	1	Check volume
PLAT151	1	Check for s.u. on volume
PLAT152	1	Check for consistency of s.u. on volume and cell parameters
PLAT155	4	Check for reduced cell *aP*
PLAT156	4	Check for non-standard axial order
PLAT157	4	Check for non-standard monoclinic β angle < 90 degrees
PLAT161	4	Missing *x*-coordinate s.u.
PLAT162	4	Missing *y*-coordinate s.u.
PLAT163	4	Missing *z*-coordinate s.u.
PLAT164	4	Check for refined C—H H atoms
PLAT165	3	Check for *R*-flagged non-H atoms
PLAT166	4	Check for calc flagged atoms with s.u.'s on coordinates
PLAT170	4	Check for sufficient data in atom data loop
PLAT199	1	Test for *SHELXL* room-temperature default (cell)
PLAT200	1	Test for *SHELXL* room-temperature default (data collection)
PLAT201	2	Test for isotropic non-H atoms in main residue(s)
PLAT202	3	Test for isotropic non-H atoms in anion? or solvent?
PLAT210	3	Test for all-isotropic a.d.p.(s)
PLAT211	2	Test for NPD a.d.p.'s (1.0) in main residue(s)
PLAT212	2	Test for NPD a.d.p.'s in anion? & solvent? [0, 1]
PLAT213	2	Test ratio a.d.p. max/min in main residue(s)
PLAT214	2	Test ratio a.d.p. max/min in anion? or solvent?
PLAT215	3	Test for unusual disordered atom a.d.p. in main residue
PLAT216	3	Test for unusual disordered atom a.d.p. in minor residue
PLAT217	1	Test for incomplete U_{ij} data
PLAT220	2	Test $U_{\rm eq}$(max)/$U_{\rm eq}$(min) range for non-H atoms in non-solvent
PLAT221	4	Test $U_{\rm eq}$(max)/$U_{\rm eq}$(min) range for non-H atoms in solvent
PLAT222	3	Test $U_{\rm eq}$(max)/$U_{\rm eq}$(min) range for H atoms in non-solvent
PLAT223	4	Test $U_{\rm eq}$(max)/$U_{\rm eq}$(min) range for H atoms in solvent
PLAT230	2	Hirshfeld rigid-bond test [*Acta Cryst.* (1976), A**32**, 239–244]
PLAT231	4	Hirshfeld rigid-bond test [*Acta Cryst.* (1976), A**32**, 239–244]
PLAT232	2	Hirshfeld rigid-bond test (metal-*X*) [*Acta Cryst.* (1976), A**32**, 239–244]
PLAT233	4	Hirshfeld rigid-bond test (metal-*X*) [*Acta Cryst.* (1976), A**32**, 239–244]
PLAT241	2	Test for unusually high $U_{\rm eq}$ as compared with bonded neighbours
PLAT242	2	Test for unusually low $U_{\rm eq}$ as compared with bonded neighbours
PLAT243	4	Test for unusually high solvent $U_{\rm eq}$ as compared with bonded neighbours
PLAT244	4	Test for unusually low solvent $U_{\rm eq}$ as compared with bonded neighbours
PLAT250	2	Test for unusual anisotropic average U_{ij}
PLAT301	3	Test for main residue(s) disorder
PLAT302	4	Test for (anion/solvent) disorder
PLAT305	2	Test for isolated hydrogen atoms
PLAT306	2	Test for isolated oxygen atoms
PLAT307	2	Test for isolated metal atoms
PLAT308	2	Test for single-bonded metal atoms
PLAT309	2	Test for single-bonded oxygen atoms
PLAT310	2	Test for 'too close' (symmetry-related) full-weight atoms
PLAT311	2	Test for isolated disordered oxygen atoms
PLAT312	2	Test for C=O—H
PLAT313	2	Test for O with three covalent bonds
PLAT318	2	Hybridization problem on N in main residue(s)
PLAT319	2	Hybridization problem on N in solvent/ion

Table A5.7.2.2 (*cont.*).

Test name	Type	Purpose
PLAT320	2	Hybridization problem on C in main residue(s)
PLAT321	2	Hybridization problem on C in solvent/ion
PLAT322	2	Hybridization problem on non-C in main residue(s)
PLAT323	2	Hybridization problem on non-C in solvent/ion
PLAT324	2	Check for possibly missing H on coordinating *X*—N—*X* in main residue
PLAT325	2	Check for possibly missing H on coordinating *X*—N—*X* in solvent/anion
PLAT326	2	Check for possibly missing hydrogen atom on carbon with sp^3-like geometry in the main residue
PLAT327	2	Check for possibly missing hydrogen atom on carbon with sp^3-like geometry in the solvent/anion
PLAT328	2	Check for possibly missing H on potentially sp^3 P
PLAT330	2	Check average phenyl C—C
PLAT331	2	Check average phenyl C—C
PLAT332	2	Check phenyl C—C range
PLAT333	2	Check average in multiply substituted benzene C—C
PLAT334	2	Check average in multiply substituted benzene C—C
PLAT335	2	Check multiply substituted benzene C—C range
PLAT338	2	Check average torsion angle in cyclohexane ring
PLAT340	3	Check bond precision for C—C in light-atom structures [$Z(\max) < 20$]
PLAT341	3	Check bond precision for C—C in structures [$19 < Z(\max) < 40$]
PLAT342	3	Check bond precision for C—C in structures [$Z(\max) > 39$]
PLAT350	3	Test for short C—H (ångstrom difference) XRAY: 0.96
PLAT351	3	Test for long C—H (ångstrom difference) XRAY: 0.96
PLAT352	3	Test for short N—H (ångstrom difference) XRAY: 0.87
PLAT353	3	Test for long N—H (ångstrom difference) XRAY: 0.87
PLAT354	3	Test for short O—H (ångstrom difference) XRAY: 0.82
PLAT355	3	Test for long O—H (ångstrom difference) XRAY: 0.82
PLAT360	2	Test for short C4—C4 (ångstrom difference) XRAY: 1.54
PLAT361	2	Test for long C4—C4 (ångstrom difference) XRAY: 1.54
PLAT362	2	Test for short C4—C3 (ångstrom difference) XRAY: 1.52
PLAT363	2	Test for long C4—C3 (ångstrom difference) XRAY: 1.52
PLAT364	2	Test for short C4—C2 (ångstrom difference) XRAY: 1.46
PLAT365	2	Test for long C4—C2 (ångstrom difference) XRAY: 1.46
PLAT366	2	Test for short C?—C? (ångstrom difference) XRAY: 1.50
PLAT367	2	Test for long C?—C? (ångstrom difference) XRAY: 1.50
PLAT368	2	Test for short C3—C3 (ångstrom difference) XRAY: 1.34
PLAT369	2	Test for long C3—C3 (ångstrom difference) XRAY: 1.34
PLAT370	2	Test for short C3—C2 (ångstrom difference) XRAY: 1.31
PLAT371	2	Test for long C3—C2 (ångstrom difference) XRAY: 1.31
PLAT372	2	Test for short C2—C2 (ångstrom difference) XRAY: 1.25
PLAT373	2	Test for long C2—C2 (ångstrom difference) XRAY: 1.25
PLAT374	2	Test for long N—N bond (> 1.45 Å)
PLAT380	4	Test for incorrectly oriented methyl moiety
PLAT390	3	Test methyl moiety *X*—C—H bond angle
PLAT391	3	Test methyl moiety H—C—H bond angle
PLAT395	2	Test *X*—O—*Y* angle
PLAT396	2	Test Si—O—Si angle
PLAT410	2	Test for short non-bonding intra H···H contacts
PLAT411	2	Test for short non-bonding inter H···H contacts
PLAT412	2	Test for short non-bonding intra H···H contacts (involving *X*H3)
PLAT413	2	Test for short non-bonding inter H···H contacts (involving *X*H3)
PLAT414	2	Test for short non-bonding intra *D*—H···H—*X* contacts
PLAT415	2	Test for short non-bonding inter *D*—H···H—*X* contacts
PLAT416	2	Test for short non-bonding intra *D*—H···H—*D* contacts
PLAT417	2	Test for short non-bonding inter *D*—H···H—*D* contacts
PLAT420	2	Test for *D*—H without acceptor
PLAT430	2	Test for short non-bonding inter $D \cdots A$ contacts
PLAT431	2	Test for short non-bonding inter HL$\cdots A$ contacts (HL = halogen)
PLAT432	2	Test for short non-bonding inter $X \cdots Y$ contacts
PLAT433	4	Test for short non-bonding minor···minor inter $X \cdots Y$ contacts
PLAT480	4	Test for too large H$\cdots A$
PLAT481	4	Test for too large $D \cdots A$
PLAT482	4	Test for too small *D*—H$\cdots A$ angle
PLAT601	2	Test for solvent accessible voids
PLAT602	4	Test for too large solvent accessible voids
PLAT603	4	Test for too large unit cell for void search
PLAT604	4	Test for too many voids
PLAT701	1	Test for consistency of bonds and coordinates in CIF
PLAT702	1	Test for consistency of angles and coordinates in CIF
PLAT703	1	Test for consistency of torsions and coordinates in CIF
PLAT704	1	Test for consistency of contact distances and coordinates in CIF
PLAT705	1	Test for consistency of H-bond *D*—H distances and coordinates in CIF
PLAT706	1	Test for consistency of H-bond H$\cdots A$ distances and coordinates in CIF
PLAT707	1	Test for consistency of H-bond $D \cdots A$ distances and coordinates in CIF
PLAT708	1	Test for consistency of H-bond *D*—H$\cdots A$ angles and coordinates in CIF
PLAT710	4	Test for linear torsions in CIF
PLAT711	1	Test for label problem for bonds in CIF
PLAT712	1	Test for label problem for angles in CIF
PLAT713	1	Test for label problem for torsions in CIF
PLAT714	1	Test for label problem for contact distances in CIF
PLAT715	1	Test for label problem for H-bond *D*—H distances in CIF
PLAT716	1	Test for label problem for H-bond H$\cdots A$ distances in CIF
PLAT717	1	Test for label problem for H-bond $D \cdots A$ distances in CIF

Table A5.7.2.2 (*cont.*).

Test name	Type	Purpose
PLAT718	1	Test for label problem for H-bond *D*—H$\cdots A$ angles in CIF
PLAT720	4	Test for unusual labels
PLAT721	1	Test for consistency of bonds and coordinates in CIF
PLAT722	1	Test for consistency of angles and coordinates in CIF
PLAT723	1	Test for consistency of torsions and coordinates in CIF
PLAT724	1	Test for consistency of contact distances and coordinates in CIF
PLAT725	1	Test for consistency of H-bond *D*—H distances and coordinates in CIF
PLAT726	1	Test for consistency of H-bond H$\cdots A$ distances and coordinates in CIF
PLAT727	1	Test for consistency of H-bond $D \cdots A$ distances and coordinates in CIF
PLAT728	1	Test for consistency of H-bond *D*—H$\cdots A$ angles and coordinates in CIF
PLAT731	1	Test for consistency of bond s.u.'s and coordinate s.u.'s in CIF
PLAT732	1	Test for consistency of angle s.u.'s and coordinate s.u.'s in CIF
PLAT733	1	Test for consistency of torsion s.u.'s and coordinate s.u.'s in CIF
PLAT734	1	Test for consistency of contact distance s.u.'s and coordinate s.u.'s in CIF
PLAT735	1	Test for consistency of H-bond *D*—H distance s.u.'s and coordinate s.u.'s in CIF
PLAT736	1	Test for consistency of H-bond H$\cdots A$ distance s.u.'s and coordinate s.u.'s in CIF
PLAT737	1	Test for consistency of H-bond $D \cdots A$ distance s.u.'s and coordinate s.u.'s in CIF
PLAT738	1	Test for consistency of H-bond *D*—H$\cdots A$ angle s.u.'s and coordinate s.u.'s in CIF
PLAT741	1	Test for missing bond s.u. in CIF
PLAT742	1	Test for missing angle s.u. in CIF
PLAT743	1	Test for missing torsion s.u. in CIF
PLAT744	1	Test for missing contact distance s.u. in CIF
PLAT745	1	Test for missing H-bond *D*—H distance s.u. in CIF
PLAT746	1	Test for missing H-bond H$\cdots A$ distance s.u. in CIF
PLAT747	1	Test for missing H-bond $D \cdots A$ distance s.u. in CIF
PLAT748	1	Test for missing H-bond *D*—H$\cdots A$ angle s.u. in CIF
PLAT751	4	Test for senseless bond s.u. in CIF
PLAT752	4	Test for senseless angle s.u. in CIF
PLAT753	4	Test for senseless torsion s.u. in CIF
PLAT754	4	Test for senseless contact distance s.u. in CIF
PLAT755	4	Test for senseless H-bond *D*—H distance s.u. in CIF
PLAT756	4	Test for senseless H-bond H$\cdots A$ distance s.u. in CIF
PLAT757	4	Test for senseless H-bond $D \cdots A$ distance s.u. in CIF
PLAT758	4	Test for senseless H-bond *D*—H$\cdots A$ angle s.u. in CIF
PLAT761	1	Test for the presence of at least one *X*—H in the CIF
PLAT762	1	Test for at least one *X*—*Y*—H or H—*Y*—H entry in the CIF
PLAT763	1	Test for missing bonds in CIF
PLAT764	4	Test for overcomplete bonds in CIF
PLAT770	2	Test for suspect C—H bonds in CIF (not caught otherwise)
PLAT771	2	Test for suspect N—H bonds in CIF (not caught otherwise)
PLAT772	2	Test for suspect O—H bonds in CIF (not caught otherwise)
PLAT773	2	Test for suspect C—C bonds in CIF (not caught otherwise)
PLAT779	2	Test for suspect angle in CIF (not caught otherwise)
PLAT780	2	Test whether coordinates form a connected set
PLAT790	4	Test whether c.g. residue in unit-cell box
PLAT798	4	Test for alphanumeric label on coordinate record
PLAT799	4	Test for alphanumeric label on displacement-parameter record
PLAT801	4	Test for missing, incomplete or out-of-order cell data
PLAT802	1	Test for input lines longer than 80 characters
PLAT803	1	Test for loop problem in CIF-read
PLAT804	4	Test for ARU-pack problem in *PLATON*
PLAT805	4	Test for insufficient coordinate data
PLAT806	4	Test for insufficient U_{ij} data
PLAT810	4	Test for out-of-memory problem
PLAT850	4	Test for BASF/TWIN problem in *SHELXL*

References

Adobe Systems Incorporated (1999). *PostScript language reference.* 3rd ed. Reading, MA: Addison-Wesley Longman.

Adobe Systems Incorporated (2004). *PDF reference*. 5th ed. *Adobe Portable Document Format.* Version 1.6. http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf.

Allen, F. H., Johnson, O., Shields, G. P., Smith, B. R. & Towler, M. (2004). *CIF applications. XV. enCIFer: a program for viewing, editing and visualizing CIFs. J. Appl. Cryst.* **37**, 335–338.

Allen, F. H., Kennard, O., Motherwell, W. D. S., Town, W. G., Watson, D. G., Scott, T. J. & Larson, A. C. (1974). *The Cambridge Crystallographic Data Centre, part 3. The unique molecule program. J. Appl. Cryst.* **7**, 73–78.

Becker, E. D. (2001). *Secretary General's Report. Chem. Int.* **23**, 135.

Bernstein, H. J. (1998). *cif2cif. CIF copy program.* http://www.iucr.org/iucr-top/cif/software/ciftbx/cif 2cif.src/.

Bernstein, H. J. (2000). *xmlCIF: a proposal for faithful representation of Extensible Markup Language (XML) documents within Crystallographic Information File (CIF) data sets.* http://www.bernstein-plus-sons.com/software/xmlCIF/.

Boulay, D. J. du & Hall, S. R. (1996). *PREPUB. Pre-publication tests on CIF structural data.* http://xtal.sourceforge.net/man/prepub-desc.html.

Brown, I. D. (1983). *The standard crystallographic file structure. Acta Cryst.* A**39**, 216–224.

Brown, I. D. (1988). *Standard Crystallographic File Structure-87. Acta Cryst.* A**44**, 232.

Bruno, I. J., Cole, J. C., Edgington, P. R., Kessler, M. Macrae, C. F., McCabe, P., Pearson, J. & Taylor, R. (2002). *New software for searching the Cambridge Structural Database and visualizing crystal structures. Acta Cryst.* B**58**, 389–397.

Freed, N. & Borenstein, N. (1996). *Multipurpose Internet Mail Extensions (MIME) part two: media types.* Internet Engineering Task Force. Request for comment 2046. http://www.ietf.org/rfc/rfc2046.txt

Gabe, E. J., Le Page, Y., Charland, J.-P., Lee, F. L. & White, P. S. (1989). *NRCVAX – an interactive program system for structure analysis. J. Appl. Cryst.* **22**, 384–387.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R., du Boulay, D. J. & Olthof-Hazekamp, R. (2000). *Xtal crystallographic software*. http://xtal.sourceforge.net.

Hall, S. R. & Sievers, R. (1993). *CIF applications. I. QUASAR: for extracting data from a CIF. J. Appl. Cryst.* **26**, 469–473.

Hester, J. R. & Hall, S. R. (1996). *BUNYIP: in search of errant symmetry. J. Appl. Cryst.* **29**, 474–478.

Knuth, D. E. (1986). *The TEXbook. Computers and typesetting*, Vol. A. Reading, MA: Addison-Wesley.

Le Page, Y. (1988). *MISSYM1.1 – a flexible new release. J. Appl. Cryst.* **21**, 983–984.

Murray-Rust, P. & Rzepa, H. S. (1999). *Chemical markup, XML and the Worldwide Web. 1. Basic principles. J. Chem. Inf. Comput. Sci.* **39**, 928–942.

Murray-Rust, P. & Rzepa, H. S. (2001). *Chemical markup, XML and the Worldwide Web. 2. Information objects and the CMLDOM. J. Chem. Inf. Comput. Sci.* **41**, 1113–1123.

Nardelli, M. (1983). *PARST. A system of FORTRAN routines for calculating molecular structure parameters from results of crystal structure analyses. Comput. Chem.* **7**, 95–98.

Rzepa, H. S., Murray-Rust, P. & Whitaker, B. J. (1998). *The application of chemical Multipurpose Internet Mail Extensions (chemical MIME) internet standards to electronic mail and world-wide web information exchange. J. Chem. Inf. Comput. Sci.* **38**, 976–982.

Sheldrick, G. M. (1976). *SHELX76. Program for crystal structure determination*. University of Cambridge, England.

Sheldrick, G. M. (1997). *SHELX97. Program for the refinement of crystal structures.* University of Göttingen, Germany. http://shelx.uni-ac.gwdg.de/SHELX/.

Spek, A. L. (1990). *PLATON, an integrated tool for the analysis of the results of a single crystal structure determination. Acta Cryst.* A**46** (Suppl.), C34.

Spek, A. L. (2003). *Single-crystal structure validation with the program PLATON. J. Appl. Cryst.* **36**, 7–13.

Westrip, S. P. (2004). *printCIF for Word*. http://www.iucr.org/iucr-top/cif/software/printCIFforWord/index.html.

Willis, A. C., Beckwith, A. L. J. & Tozer, M. J. (1991). *trans-3-Benzoyl-2-tert-butyl-4-isobutyl-1,3-oxazolidin-5-one. Acta Cryst.* C**47**, 2276–2277.

Subject index

Ab initio structure determination, 123
Abbreviations within CIF data names, 85
Absolute configuration, 48, 107, 171, 173
Absorption correction, 98
Acta Crystallographica, 5, 6, 21, 24, 54, 114, 128, 502, 507, 509, 512, 513, 557–562, 564–566
ADIT, 197, 541–543
Alias, *see* data name
Ambient conditions
  coreCIF, 93
  mmCIF, 149
  pdCIF, 118, 126
American Society for Testing and Materials, 44
Analyser, 94, 95, 127, 203
Aperiodic structures, 131–140
API, 484, 494, 522, 523, 541, 544
Apparatus
  coreCIF, 94–95
  pdCIF, 119–120
Application programming interface, *see* API
Archiving, 13, 20, 22, 101, 112, 481
Area detectors, 24, 37, 117, 141, 558
  resolution of, 95
Array data, 199–201
  encoding, 200, 547–549
Array detector, 199
ASCII, 4, 8, 14, 16, 17, 21, 22, 24, 27, 30, 37, 39–42, 45, 114, 501, 508
ASER, *see* data exchange standards
ASN.1, *see* data exchange standards
Atom properties
  coreCIF, 102–111
  formal charge, 111
  MIF, 45, 47
  mmCIF, 164–190
  msCIF, 134–136
  rhoCIF, 141–143
Atom sites
  alternative category key (coreCIF), 103
  alternative conformations, 168–169
  anisotropic displacement parameters, 102, 103, 166
  atomic displacement parameters, 104
  attached hydrogen atoms, 105
  coreCIF, 102
  isotropic displacement parameters, 103, 166
  label, 104–105, 146, 165–166
  mmCIF, 164–167
  occupancy, 103, 104, 135, 147
  positional coordinates, 102–104, 135, 166
  refinement conditions, 105
  symmetry, 103
  thermal parameters (*see also* anisotropic displacement parameters, atomic displacement parameters, isotropic displacement parameters), 135
  transformation matrix, 103, 106
Attenuator, 94
  scaling factor, 93
Attributes, data, *see* DDL1, DDL2
Audit trail, *see* data block, history
AutoDep Input Tool, *see ADIT*
Axes
  atom-centred electron density, 141, 142
  detector, 203
  diffraction scan, 205
  goniometer, 204
  laboratory coordinate system, 95, 199, 201–202, 204, 551
  rocking, 127
  space-group setting, 206
Babel, 483
Background intensities (pdCIF), 122, 127, 128
Backslash, use in line-folding protocol, 34, 35
Backus–Naur form
  CIF syntax, 28–32, 517
  STAR File syntax, 14, 16–19, 497
BCCAB, *see* data exchange standards
Beam crossfire, 204
beCIF, 506
Bibliographic information, 113
Bibliographic references, *see* citation
Binary data, 8, 24, 37–41, 199
Binary data section (CBF), 39–42, 547, 549
Binary format, 8, 37, 483
Binary word alignment (CBF), 40
Biological function, 145
BioMagResBank, 9, 197, 494, 496, 497
BioSync, 541
bison, 485, 486, 494, 520
Block code, 15, 16, 21, 25, 46, 491
  construction guidelines, 115
  length, 23, 27
BNF, *see* Backus–Naur form
Bond angle, 109, 171, 173
Bond distance, 109, 111, 171, 173
Bond order, 7, 47, 48, 171, 173
Bond properties
  coreCIF, 108
  MIF, 45
Bond type, 44, 47, 51, 108, 171
Bond valence, 111, 190
Bonding conventions (MIF), 48
Bragg *R* factor, 100
Branching request, 492, 493
Bravais type, 207
BuildSiv, 520–522
BUNYIP, 560
Byte order (CBF), 37, 42
Byte padding (CBF), 42

C, 483, 484, 494, 495, 501, 519, 520, 526, 544
C++, 484, 495, 496, 526, 541
Calibration, 126, 200
Cambridge Crystallographic Data Centre, 3, 481, 482, 504, 560
Cambridge Structural Database, 3, 171, 482
Case sensitivity, 17, 18, 21, 23, 24, 26, 27, 511, 538
CASTOR, 557
Category, 33, 54, 62, 64, 76, 80, 90
  attribute of data definition (DDL1), 55–57, 59
  attribute of data definition (DDL2), 63, 65, 83
  definition (DDL2), 68, 80
  description (DDL1), 77
  diverse contents of category in pdCIF, 117
  equivalence to table (DDL2), 62
  equivalent item in multiple categories, 65, 66
  examples as attribute of category definition (DDL2), 69
  overview (DDL1), 77
  related, 62, 65, 69, 82
  uniqueness within data block, 69
Category group, 62, 69, 80, 90
  attribute of category definition (DDL2), 69
  comparison across CIF dictionaries, 89–91
  membership as attribute of data definition (DDL2), 64
  mmCIF, 147
Category key, 82
  absence from coreCIF symmetry categories, 111
  alternative, 103
  alternative (PD_DATA), 121
  attribute of category definition (DDL2), 64, 69
Category key
  attribute of data definition (DDL2), 80
  composite, 81
  implicit components in geometry categories, 109
  uniqueness, 62, 81
CBF, 8, 483, 488
  complement to CIF, 37
  data type, 40
  deprecated practice, 43
  experimental measurements, 199–205
  isomorph of imgCIF, 8, 37–39
  lossless image format, 37
  ordering of data items, 37, 38, 40
  software library, 544–556
  specification, 37–43
  syntax, 39, 40
  tree representation, 544–548
  use of imgCIF dictionary, 199–205
CBFlib, 484, 485, 544–556
cciflib, 484
CCP4, 484, 524, 526, 549, 552
Cell, *see* unit cell
Cell subsystem (composite structures), 133, 135
CGI (Common Gateway Interface), 541
Character set
  allowed in CIF, 22, 24, 27, 29–31, 114
  allowed in STAR, 16
Charge density, *see* electron density
checkcif, 561–562, 566
Chemical Abstracts Service, 44, 107
Chemical bonding, 141
Chemical components (mmCIF), 169–172
Chemical diagram, 558
  two-dimensional (coreCIF), 107, 558
  two-dimensional (MIF), 44, 45, 47
Chemical entity
  coreCIF, 108
  MIF description, 45
  mmCIF, 169–176
Chemical formula, 107
Chemical graph (MIF), 44
Chemical Information System, NIH/EPA, 481, 482
Chemical Markup Language, *see* CML
Chemical species, handling in mmCIF, 169–176
Chemical structure, *see also* molecular structure
  coreCIF, 108
  graph theory, 44
  MIF, 44–52
  three-dimensional (MIF), 44
Chemical Structure Association, 44
Chemistry, 7, 105, 145
  coreCIF, 102–111
  mmCIF, 164–190
  msCIF, 134–136
  pdCIF, 123
Chiral volume, 171, 173
CIF
  application of STAR File, 14, 20, 488
  archival format, 2
  article format, 557–560
  data model, 2, 7
  data-exchange mechanism, 2
  example file, 5, 6
  file transformation software, 509–515
  formal grammar, 28–32
  formal specification document, 25–36
  history, 2–10
  implementation restrictions, 27
  integration of data and experimental metadata, 92
  semantics, 23–25, 32–36
  software design considerations, 481–487, 535–538

CIF
  specification, 20–36
  standard ontology in structural science, 483
  supplementary material, 558, 562
  syntax, 5, 6, 21–23, 25–32, 117, 499
  syntax checker, 499–501, 507
  terminology, 20–21, 25
  use in publication, 557–569
  version identification, 21, 29, *see also* magic number
CIF++, 496
cif2cbf, 555, 556
cif2cif, 483, 486, 509, 511–512, 526, 558
cif2pdb, 483, 526
CIF2SX, 483
cif2xml, 483
cifdiff, 519
CIFEDIT, 504, 557
CIFIO, 526
CIFLIB, 484, 496, 522
CIFOBJ, 496, 497, 522, 540, 542
CIFPARSE, 484, 544
CIFPARSE-OBJ, 522, 540
CifSieve, 484, 486, 520–522
CIFtbx, 55, 143, 484, 508, 511, 512, 526–538
ciftex, 483, 512–515, 560
CIFTr, 197, 484
cifZinc, 518
Citation
  bibliographic references, 83
  of database entries, 113, 191
  of scientific literature, 112, 190
  of software, 113, 191
Class, *see* object-oriented STAR programming
CML, 9, 483, 496, 564
Collimation, 94, 95, 126, 203
Colour
  atom and bond representations (MIF), 47
  crystal, 98, 118
  sample (pdCIF), 118
COMCIFS, 6, 8, 20, 23, 73, 75, 86
  dictionary approval procedure, 74, 75
Comment, 16–18, 21, 27, 29–32, 34, 37, 38, 40, 45
  legitimate white space, 17
Commission on Aperiodic Crystals, 8, 131
Commission on Charge, Spin and Momentum Densities, 8, 141
Commission on Crystallographic Computing, 3, 5, 6
Commission on Crystallographic Data, 3, 5, 6
Commission on Crystallographic Nomenclature, 103
Committee for the Maintenance of the CIF Standard, *see* COMCIFS
Completeness, 94
Compliance, 25, 35, 53, 85
Composite structures, *see* incommensurate structures
Compression, image data, 200, 549, 552
Compression scheme, 43
Computer language, 484
Computer programs
  ADIT, 197, 541–543
  Babel, 483
  beCIF, 506
  bison, 485, 486, 494, 520
  BuildSiv, 520–522
  BUNYIP, 560
  CASTOR, 557
  CBFlib, 484, 485, 544–556
  cciflib, 484
  CCP4, 484, 524, 526, 541, 549, 552
  checkcif, 561–562, 566
  CIF++, 496
  cif2cbf, 555, 556
  cif2cif, 483, 486, 509, 511–512, 526, 558
  cif2pdb, 483, 526
  CIF2SX, 483
  cif2xml, 483
  cifdiff, 519
  CIFEDIT, 504, 557
  CIFIO, 526
Computer programs
  CIFLIB, 484, 496, 522
  CIFOBJ, 496, 497, 522, 540, 542
  CIFPARSE, 484, 544
  CIFPARSE-OBJ, 522, 540
  CifSieve, 484, 486, 520–522
  CIFtbx, 55, 143, 484, 508, 511, 512, 526–538
  ciftex, 483, 512–515, 560
  CIFTr, 197, 484
  cifZinc, 518
  convert_image, 555
  CYCLOPS, 508–509, 512, 526
  Democritos, 496
  DIFRAC, 483
  emacs, 499
  enCIFer, 499, 501–504, 557, 558
  flex, 486, 494, 520
  GSAS2CIF, 128, 129
  HICCuP, 504–506
  img2cif, 555
  JANA2000, 132, 135, 136
  Jumbo, 496, 497
  lex, 486
  makecbf, 555
  MAXIT, 484
  Mercury, 562
  MISSYM, 560
  mmCIF Loader, 542
  mmLib, 523–525
  mmView, 525
  MOSFLM, 202
  NRCVAX, 560
  OOSTAR, 484, 495, 496
  OpenMMS, 522–523, 541
  ORFLS, 3
  PARST, 560
  pdb2cif, 483, 526
  PDB_EXTRACT, 197, 483, 542, 543
  pdCIFplot, 128
  PREPUB, 560, 566
  printcif, 559, 560
  printCIF for Word, 506–507, 558, 560
  PROTIN/PROLSQ, 161
  PyCifRW, 517
  QUASAR, 14, 483, 486, 509–512, 514, 558
  RESTRAIN, 161
  SHELX76, 557
  SHELX97, 558
  SHELXL, 483
  Star_Base, 20, 484, 491–494, 496, 497, 499, 511
  StarDOM, 497
  starlib, 494, 497
  StarMarkUp, 494, 495
  STAR::Parser, 515–517
  Star.vim, 494, 499
  vcif, 499–501
  vim, 494, 499
  XD, 143
  XRAY, 3
  Xtal, 560
  yacc, 486, 520
  Yapps2, 517
  zb, 519
  ZINC, 518–519
  zincCif, 519
  zincGrep, 519
  zincNl, 519
  zincSubset, 519
Conditional request, 492, 493
Conformational representation, using save frame (MIF), 46
Connectivity
  coreCIF, 102, 105–108, 558
  MIF, 45, 47
  structural interactions (mmCIF), 182
Content handler, 24, 34
Content-Encoding (MIME header), 24
Content-Transfer-Encoding (MIME header), 39, 41, 42
Content-Type (MIME header), 24, 39, 41
Context, within data structure, *see also* STAR File, data instances and context; scope, 20, 488–491
Context-sensitive productions (formal grammar), 29, 31
Continuous-wavelength neutron experiments, 8
Control character, 30
Controlled vocabulary, 9, 32
Conventions, alternative
  bonding (MIF), 48
  units (CIF), deprecated, 33, 79
convert_image, 555
Corba, 523, 541
Core CIF (core Crystallographic Information File) *see* coreCIF
coreCIF
  atom properties, 102–111
  category grouping, 89–91
  category groups, 92
  chemistry, 102–108
  classification and use, 92–116
  data collection, 93–97
  dual formalism, 92
  embedded in mmCIF dictionary, 76, 92, 102, 144
  experimental measurements, 92–98
  history, 7
  list of categories, 116
  refinement, 98–101
  relationship to symCIF, 8, 110
  structural model, 102–111
  structure analysis, 98–102
Crambin (mmCIF example structure), 145–147
Crenel function, *see* special modulation functions
Cruickshank diffraction-component precision index, 159
Cryo-electron microscopy, 197
Crystal
  characterization, 98, 151
  colour, 98, 118
  faces, 98, 134
  growth, 151–152
  system, 110, 111
  treatment, 94
  type of aperiodicity, 134
Crystallization, *see* crystal, growth
Crystallographic Binary File, *see* CBF
Crystallographic Information File, *see* CIF
CYCLOPS, 508–509, 512, 526
CYCLOPS2, *see CYCLOPS*

`data_` (STAR File keyword), 16, 21, 22, 25, 26, 31, 32, 40
Data archiving, 2, 3
Data array, 199–201
Data block, 14–18, 20, 21, 25, 26, 32, 38, 40, 45, 46, 62, 70, 123, 489–491, 493–495, 546, 548
  container for published crystal structure, 558
  data definition block (DDL1), 54, 56, 76
  history, 114, 194
  intra-file relationship, 25, 114, 115, 117, 123, 124, 129, 132, 136, 139, 194–195
  publication text, 558
  purpose, 115
  reference, 115, 123, 124, 136
  unique identifier (msCIF), 139
  unique identifier (pdCIF), 123
  uniqueness, 21, 115
Data collection
  CBFlib high-level functions, 549–552
  coreCIF data items, 93–97
  imgCIF data items, 202–204
  mmCIF data items, 149–151
  msCIF data items, 133
  pdCIF data items, 120–121
Data definition, *see also* DDL1, DDL2, dictionary
  attribute of data definition (DDL1), 55, 58, 78
  attribute of data definition (DDL2), 63, 66
  DDL1, 55–58, 77–79

Data definition
  granularity, 83
  guidelines, 77–79, 81–85
Data dictionary, *see* dictionary
Data exchange standards, *see also* CBF, CIF, MIF, STAR File
  ASER, 3
  ASN.1, 5, 14, 44, 51, 483
  BCCAB, 3–5
  card-image format, 3
  fixed format, 3, 13, 482
  HDF, 126, 483
  HGML, 14
  JCAMP-DX, 5
  MTZ, 524
  NeXus, 126, 483
  PDB, 3, 5, 14, 104, 144, 193, 482, 484, 524, 539
  SCFS, 3, 4, 45, 557
  SMD, 5, 44, 45
  SMILES, 51
  Z39.50, 5
Data item, 15, 20, 25, 32, 40, 45, 488, 491, 492, 494
  attributes, *see* DDL1, DDL2
  context, 488–491, 511, 519
  lowest level of organization (DDL2), 62
  mandatory membership of category (DDL2), 65
  matrix or vector structure (DDL2), 67
  order independence, 20, 23, 26, 53, 481, 483, 485, 488, 526
  ordering in CBF, 37, 38, 40
  relationships, 32, 33, 53, 54, 56, 57, 59, 61, 64–67, 76, 79, 82
  uniqueness, 46
Data loop list, *see* looped list
Data model, 13, 488, 491
  choice of DDL, 75, 76
  CIF, 2, 7
  hierarchical, 61, 488, 495
  object-oriented, 54, 61, 488, 495–497
  relational, 54, 61, 75, 495, 497
Data name, 15, 18, 20, 21, 25, 26, 40, 45, 54
  alias, 32, 54, 61, 63, 66, 144, 529
  attribute of data definition (DDL1), 55, 58
  attribute of data definition (DDL2), 63, 65
  case insensitivity, 21, 526, 538
  consistency of abbreviations, 85
  construction, 23, 32, 76, 77, 79, 84
  correspondence between coreCIF and mmCIF, 144
  decomposition into category and item components (DDL2), 33, 65, 79, 83, 84
  length, 22, 23, 27
  local, 33, 74
  semantics, 23, 84
  uniqueness, 16, 22, 25, 26, 45, 558
  validation, 507–509
Data packet, *see* loop packet
Data request, 491, 492
Data set, 16, 117, 123, 544–546
Data structure, 13, 127, 482–485, 488, 495, 520, 523, 544–546, 552
Data type, 23, 33, 53, 76, 82, 492, 520
  attribute of data definition (DDL1), 55, 58, 77
  attribute of data definition (DDL2), 63, 67, 83
  CBF, 40
  character, 22, 24, 33
  construction (DDL1), 59, 85
  construction (DDL2), 68
  extended types, 24, 34, 67, 68, 85
  extended types (DDL2), 23, 64
  handling in CBF, 548
  inferred, 34
  null, 24, 33
  numeric, 22, 23, 26, 33
  primitive, 22, 23, 33, 68
  standard uncertainty, 23
  uchar, 24, 33
Data validation tests, *see also checkcif*, 561–562, 566–568
Data value, 13, 15, 20, 21, 25, 26, 45, 492
  alternative values, 50
  default, 24, 56–58, 66, 94
  enumeration as attribute of data definition (DDL1), 53, 55, 57, 58, 78
  enumeration as attribute of data definition (DDL2), 67
  idempotency within save frame, 15
  inapplicable (`.`), 24, 34
  missing (`?`), 24, 34
  range as attribute of data definition (DDL2), 67
  relationships, 60
Database
  entries, 113, 186–187
  loader and support tools (mmCIF), 541–542
  query language, 50, 491
DDL, 7, 32, 47, 53, 54, 73, 208, 483, 494–496, 559
  choice of data model, 76
  role in mmCIF metadata architecture, 540
  self-defining, 53
  syntax, 55
DDL0 (undocumented DDL), 7
DDL1, 7, 8, 32–34, 54, 73, 76, 496
  application of STAR File, 14, 53
  comparison with DDL2, 55, 76
  descriptive attributes, 58
  dictionaries, 53, 90, 132
  dictionary, 471–472
  dictionary organization, 53–54, 76
  dictionary registration attributes, 60
  history, 54
  identification attributes, 58
  local data name, 74
  specification, 53–60
  syntax, 53
  typing attributes, 58
  use in coreCIF, 92
  use in MIF, 45
DDL2, 7, 8, 32–34, 54, 73, 76, 488, 542
  application of STAR File, 14, 61
  backward compatibility with DDL1, 61, 63
  comparison with DDL1, 55, 76
  description of categories, 68–70, 80–81
  description of data blocks, 70
  description of data items, 65–68, 81–83
  description of dictionaries, 70
  description of methods, 70
  dictionaries, 61, 90, 207
  dictionary, 473–478
  history, 54, 61, 144
  local data name, 74
  organization, 62
  presentation of definitions, 61
  self-defining, 63
  specification, 61–70
  syntax, 61
  use in core dictionary embedded in mmCIF, 92
  use in mmCIF, 144
Deadtime, 95, 126, 128, 203
Decomposition temperature, 107
Democritos, 496
Density functional theory, 141
Dictionary, 2, 6, 32, 62, 73, 494, 495
  administration of standard, 73
  augmented (*CifSieve*), 520
  authorship, 73
  browser, 502, 504
  certification, 73
  composite, 88, 141
  coreCIF, 92–116
  DDL1, 471–472
  DDL2, 473–478
  design guidelines, 73–85
  identification, 35, 70, 76, 79, 85, 89
  identification (DDL1), 55, 60, 77
  imgCIF, 199–205, 444–458
  importance of standard data names, 73
  lexicographic, 53, 55
  local, 73
Dictionary
  management of multiple dictionaries, 85–89
  merging protocol, 88–89
  MIF, 467–470
  mmCIF, 144–196
  msCIF, 131–140, 270–289
  organization, 53–54, 62, 76–77, 79–81
  PDB exchange (pdbx), 195–197
  pdCIF, 117–130, 258–269
  register, 86, 87
  revision history, 55, 70, 76, 79, 89
  rhoCIF, 141–143, 290–294
  symCIF, 206–208, 459–466
Dictionary definition language, *see* DDL
Dictionary maintenance group, 75
Diffraction experiment, overview, 93, 149
Diffraction point symmetry, 94
Diffraction scan, 199, 200, 204
  type, 95
Diffractometer, 95
  axes, 95
DIFRAC, 483
Dihedral angle, 109
Dimensionality, superspace, 131, 133, 134, 137
Discontinuous atomic domains, 131, 132, 136
Disorder, 103–106, 131, 160
Displacive modulation, 131, 135, 136, 138
Disulfide link, 145, 182
Document object model, 496, 497
Document-type definition, *see* DTD
DREL, 496
DTD, 9, 497, 562

EBCDIC, 22
EBNF (extended BNF), *see* Backus–Naur form
Efficiency, 2, 37, 38, 53
Electron density, 141–143
Electron density CIF (electron density Crystallographic Information File), *see* rhoCIF
Element symbol, 106
emacs, 499
E-mail, 4, 562
Empty file, 17
Enantiomer, 50
enCIFer, 499, 501–504, 557, 558
Encoding, 22, 24, 27, 34
Energy-dispersive X-ray experiments, 8, 117, 122, 127
Enumeration, *see* data value
Error checking, 41
Estimated standard deviation (e.s.d), *see* standard uncertainty
European Bioinformatics Institute, 195
Example
  attribute of data definition (DDL1), 58
  attribute of data definition (DDL2), 64, 67
Exchange protocol, 2
Experimental measurements
  CBF data items, 199–205
  coreCIF data items, 92–98
  imgCIF data items, 199–205
  mmCIF data items, 148–152
  msCIF data items, 133–134
  pdCIF data items, 118–121
Extensibility, 2, 13, 37, 51, 53, 54, 83
Extensible markup language, *see* XML
External reference file, 25, 32
  standard chemical data for monomers, 170
  standard instrument settings, 126
Extinction, 99

$F(000)$, 98
File extension
  CBF, 40
  imgCIF, 40
FILM, 541
flex, 486, 494, 520
Flexibility, 13, 37, 48, 51, 53, 54, 495, 526
Formula, chemical, 107
Fortran, 484, 508, 509, 519, 520, 526–538

Fourier series modulation function, 132, 135, 138, 139
Frame, image data, 199, 203, 205
Frame code, 15, 16, 25, 46, 489, 491, 495
  length, 27
  uniqueness, 25
Friedel equivalents, 101
Ftp, 4, 22, 40, 41, 86

Generator, 207
Generic structures (MIF), 7, 490
Genetically manipulated source of biological macromolecules, 174
Geometry, 108–110, 136, 171–174, 176–178
`global_` (STAR File keyword), 16, 22, 26, 27, 40, 47
Global block, 14–18, 45, 47, 49, 489–492, 494
  forbidden in CIF, 27
  handling by *CIFtbx*, 529
Goniometer, 95, 98, 202, 204, 551
Grammar, 13
Granularity of data definitions, 83
Graph theory (chemical structure), 44
Graphical editor, 501–507
Greek letters, 36
GSAS2CIF, 128, 129

Hall space-group symbol, 110, 111, 206, 207
HDF, *see* data exchange standards
Header (CBF), 37–40, 42
Hermann–Mauguin symbol, 110, 111
HGML, *see* data exchange standards
HICCuP, 504–506
Hierarchical data model (DDL2), 61, 62
Hill convention for chemical formulae, 107
HTML, 482, 483, 494, 496, 515
Human Genome Mapping Library language (HGML), *see* data exchange standards
Hydrogen bond, 109, 110, 182, 188

Image CIF (image Crystallographic Information File), *see* imgCIF
Image data, 8, 24, 37, 38, 42, 117, 199–205, 544–556
Image plate, 8, 117, 126, 127, 552
img2cif, 555
imgCIF, 127
  category grouping, 89–91
  classification and use, 199–205
  dictionary, 444–458
  experimental measurements, 199–205
  history, 8, 483
  isomorph of CBF, 8, 37–39
  potential use for publication graphics, 558
  software library, 544–556
  specification, 37–43
Incident-intensity monitor, 128
Incommensurate structures, 8, 102, 131–140, 206, 208
Index terms, 113
Indexing in powder diffraction, *see* phase identification
Inorganic structures, described by coreCIF, 92
Instrumentation
  coreCIF, 94–95
  pdCIF, 119–120
Integration time, 205
Intensity measurement, 95–97
  in pdCIF, 120–121
  using an image detector, 200, 204, 550
Intensity measurements, 149
  binning, 97, 101, 102, 127
  excluded from refinement, 100, 101, 159
  number used in refinement, 101
  processed, 96
  raw, 96
  scaling, 96–98, 101, 102, 122
  standard, 96, 97
  standard uncertainty, 99
  threshold, 99–102, 159, 163
Intensity measurements
  units (mmCIF), 163
  used in refinement, 100–101, 162–164
Interatomic contacts, 109
International Centre for Diffraction Data (ICDD), 123
International Structural Genomics Organization, 195
International Union of Crystallography, *see* IUCr
International Union of Pure and Applied Chemistry, *see* IUPAC
Interoperability, 9, 25, 61, 73
Intramolecular geometry, 109, 173
Irreducible set of data items, *see also* subcategory, 54, 56, 58, 64, 77, 79–81
Isomorphous phasing, *see* phasing
IUCr, 2, 3, 5–8, 14, 20, 45, 61, 73, 131, 141, 195, 557
IUPAC, 9, 94, 107, 564

JANA2000, 132, 135, 136
Java, 496, 497, 523
JCAMP-DX, *see* data exchange standards
Journal, *see* publishing
Jumbo, 496, 497

Key, *see* category key
Keywords, 13, 16–18, 189

LaTeX, 515
Lattice centring, 111
Laue class, 207
Least-squares plane, 109
lex, 486
Lexical scan, 29, 30, 486
Lexical tokens, 17, 486, 495
Line folding, 34–35
Line length
  CBF, 39, 40
  CIF, 22–24, 27, 34, 500, 511
  MIF, 45
Line terminator, 14, 22, 27, 30, 37, 39, 41
  CBF convention, 39, 40
  multi-platform convention, 27, 30
  STAR File, 17
List, *see also* looped list
  attribute of data definition (DDL1), 56, 59, 78
  dependencies, 56, 59, 60, 78, 81
  join (DDL1), 56
  key, 56, 59, 60, 81
  mandatory item, 56, 57, 59, 60, 78, 82
Livetime, 128, *see also* deadtime
`loop_` (STAR File keyword), 15, 16, 21, 22, 26, 31, 40, 45, 540
Loop level, 15, 58
Loop packet, 15, 45, 53, 488, 489, 493
Looped list, 15, 16, 18, 20, 21, 40, 45, 54, 78, 488, 494, 511
  editor, 503, 506
  nested, *see* nested loop
  nesting prohibited in CIF, 21, 27

Macromolecular CIF (macromolecular Crystallographic Information File), *see* mmCIF
Macromolecular structure, 145–147, 178–189
  biological units, 179–181
  secondary structure, 181–182
Magic number, 21, 29, 37, 40
Magnetic structures, 206
makecbf, 555
MAXIT, 484
Melting point, 107
Mercury, 562
Meta character, *see* data value, missing; data value, inapplicable
Metadata, 25, 62, 92, 114–116, 123, 136, 194–195, 481, 483, 496, 539, 540, 542, 562
Methods, 62, 70, 207, 208, 496, 497, 540
  attribute of category definition (DDL2), 69
  attribute of data definition (DDL2), 67
MIF
  application of STAR File, 14, 44, 45, 488
  as complement to CIF, 44
  compared with coreCIF, 108
  data model, 7
  design objectives, 45
  development, 7, 44
  dictionary, 467–470
  specification, 44–52
MIME
  candidate content headers in CIF, 24
  chemical/x-cif, 562
  Content-Encoding, 24
  Content-Transfer-Encoding, 39, 41, 42
  Content-Type, 24, 39, 41
  encoding in CBF, 37–39, 41, 43, 547, 556
  X-Binary-Element-Type, 39, 41
  X-Binary-Size, 39, 41
MISSYM, 560
mmCIF
  application of STAR File, 488
  atom properties, 164–190
  category grouping, 89–91
  category groups, 147, 148
  chemistry, 164–190
  classification and use, 144–196
  complemented by NMRStar, 9
  conversion to XML, 9
  data collection, 149–151
  design considerations, 144–145
  dictionary, 295–443
  experimental measurements, 148–152
  history, 7, 8, 144–145
  list of categories, 195–196
  rationale for use of DDL2, 61
  refinement, 158–162
  relation to PDB format, 192
  software tools, 522–525
  structural model, 145–147, 164–190
  structure analysis, 152–164
  use as domain ontology, 9
  use in PDB data management, 195–197, 539–543
mmCIF Loader, 542
mmLib, 523–525
mmView, 525
Modulated structures, *see* incommensurate structures
Modulated structures CIF (modulated structures Crystallographic Information File), *see* msCIF
Molecular averaging, *see* phasing
Molecular chemistry, differences between coreCIF and mmCIF, 169–174
Molecular connectivity, *see* connectivity
Molecular fragment represented by save frame (MIF), 46
Molecular geometry, *see* geometry
Molecular Information File, *see* MIF
Molecular structure, *see also* chemical structure
  MIF, 44–52
  three-dimensional (MIF), 45, 47
Molecule, 7
  MIF description, 45
Monochromatization, 94, 126
Monomer components of biological macromolecules, 170–174
MOSFLM, 202
Mounting, 95, 119
msCIF
  atom properties, 134–136
  category grouping, 89–91
  category groups, 131
  chemistry, 134–136
  classification and use, 131–140
  data collection, 133
  design considerations, 131–132
  dictionary, 270–289
  examples of use, 137–139

msCIF
experimental measurements, 133–134
history, 8, 131
refinement, 134
structural model, 134–136
structure analysis, 134
symmetry, 136
MTZ, *see* data exchange standards
Multi-byte character set, 22
Multichannel analyser, 127
Multiple detectors, 117, 126, 127
Multiple isomorphous replacement (MIR), *see* phasing
Multiple-wavelength anomalous dispersion (MAD), *see* phasing
Multiplicity, 103, 208
Multipole expansion model, 141, 142
Multipurpose Internet Mail Extensions, *see* MIME
Multivariate logic, 492

Name space, 33, 74
Natural tissue, source of biological macromolecules, 175
Nested loop, 15, 58, 59
image data structure in *CBFlib*, 545
MIF, 45
STAR File, 18, 58, 488, 489, 495, 497
Networking, 4
Neutron diffraction, 117, 123, 124, 126, 129
Neutron time-of-flight detection, 117, 127
NeXus, *see* data exchange standards
NMR structures, 9, 169, 197, 496, 539, 540
NMRStar, 9, 20, 197, 496, 497
Noncrystallographic symmetry, 131, 153, 160, 161, 184–186
NRCVAX, 560
Nucleic Acid Database, 496, 522

Object Management Group, 541
Object-oriented CIF libraries, 515, 517, 522, 526–538, 540
Object-oriented STAR programming, 495–497
Objective-C, 484, 495
'Observed' reflections, *see* intensity measurements, threshold
Occupancy, *see* atom sites, occupancy, 106
Occupational modulation, 131, 135, 138
Ontology, 2, 9, 32, 51, 53, 481, 483, 523, 539
OOSTAR, 484, 495, 496
OpenMMS, 522–523, 541
Operating-system dependence, 17, 22, 27, 30, 37, 40
Optical rotation, 107
Order independence, *see* data item, ordering in CBF
ORFLS, 3
Orientation matrix, 95
Origin choice, space group, 206

Packing geometry, *see* geometry
Parsing rules, 17, 21, 23, 25, 30, 81, 83, 85, 109, 114, 486, 495, 499, 517, 535
PARST, 560
PDB, *see* data exchange standards
PDB exchange data dictionary, 539–543
pdb2cif, 483, 526
PDB_EXTRACT, 197, 483, 542, 543
pdCIF
category grouping, 89–91
category groups, 118
chemistry, 123
classification and use, 117–130
data collection, 120–121
design considerations, 117
dictionary, 258–269
examples of use, 126–130
experimental measurements, 118–121
history, 8
measured intensities, 121
naming convention for data items, 117
processed intensities, 121
pdCIF
simulated intensities, 122, 128, 129
structural model, 123
structure analysis, 121–123
unprocessed measurements, 121, 126–129
pdCIFplot, 128
Peak intensities (pdCIF), 122, 123
Peak positions (pdCIF), 121–123
Peak width (pdCIF), 121, 122
Perl, 495, 515, 520
Phase description (pdCIF), 123, 124, 128, 129
Phase identification, 117, 129
Phasing, 98, 152–158
characterization of data sets, 153, 157–158
isomorphous replacement, 153–154
MAD, 154–155
MIR, 155–156
molecular averaging, 153
Phason correction, 134, 135
Pixel, 552
Pixel array, *see* data array
Pixel characteristics, 200–201
PLATON, 560, 566
Point group, 207
Polarization, 204
Portability, 2, 13, 20, 22, 23, 27, 37, 482, 535
Position-sensitive detector, 117
Powder CIF (powder Crystallographic Information File), *see* pdCIF
Powder diffraction, 8, 117–130, 558
Powder Diffraction File, 122, 123
Precision electron density studies, 141
Preferred orientation, 127
PREPUB, 560, 566
Pressure, *see* ambient conditions
printcif, 559, 560
printCIF for Word, 506–507, 558, 560
Privileged constructs, 16
Production (formal grammar), 17, 18, 29–32
Profile fitting, 122
Protein Data Bank, 2, 3, 190, 481, 482, 496, 517, 522, 539–543
data exchange dictionary, 195–197
Protein Data Bank Japan, 195
Protein Expression Purification and Crystallization database (PEPCdb), 197
Protein production, 197
PROTIN/PROLSQ, 161
Protocols
data exchange, 2, 6
dictionary merging, 35, 88–89, 141
dictionary retrieval, 87–88
http, 557
line folding, 24, 34–35
network file sharing, 27
networking, 4
Publishing, *see also Acta Crystallographica, Zeitschrift für Kristallographie*
CIF as a markup language, 483
CIF as supplementary material, 562
CIF editors, 507
contents of an article, 114
data items (coreCIF), 112–114
data items (mmCIF), 190–194
electronic submission, 5, 6, 557, 560, 562
potential use of CBF for graphics, 37
proofs, 560
request list, 558, 559, 564–566
small-molecule crystal structures, 557–569
PyCifRW, 517
Python, 495, 504, 515, 517, 524

Quantum chemistry basis sets, 488, 491, 493
QUASAR, 14, 483, 486, 509–512, 514, 558
Quasicrystals, 131, 206, 208
Query language (MIF), 50
Quoted text string, 14, 18, 22, 24, 26, 31, 33, 34, 38–41, 45, 59

R factor, 99, 158
Bragg, 100
Hamilton generalized, 160
Luzzati analysis, 160
R_{free}, 159, 164
R_{merge}, 164
real-space, 184
Racemic mixture, 50
Radiation
mean wavelength (Laue experiments), 101
polychromatic, 95, 101, 117, 122
Radiation detector, 95, 126
Radiation generator, 94, 95
Radiation source, 94, 95, 126
Radiation type, 94, 95, 120
Radiation wavelength, 94
Reaction schemes, 7
Record padding, 22
Redundancy, 29, 41, 48, 79, 101, 102, 159
Reference list, *see* citation
Refinement
characteristics of individual reflections, 100
constraints, 103, 105
coreCIF data items, 98–101
cycle, 101, 162
goodness of fit, 99
history, 159, 162
ideal groups, 174
mmCIF data items, 158–162
msCIF data items, 134
number of reflections included in, 101
of hydrogen atoms, 99
pdCIF, 122
quality metrics, 160
residual factor, *see R* factor
restraints, 103, 105, 160–161, 173
Rietveld, *see* Rietveld refinement
solvent (mmCIF), 159
structure-factor coefficient, 99
weighted residual, 99
weighting scheme, 99
Reflection table (pdCIF), 121, 123, 128, 129
Reflections, *see* intensity measurements
Register
CIF dictionaries, 86, 87, 559
reserved prefixes, 74, 559
Relational data model, *see* data model
Research Collaboratory for Structural Bioinformatics, 195, 522, 539–543
Reserved feature, 20, 27, 33
Reserved prefix (CIF data names), 33, 74, 87, 559
Reserved words (STAR), *see also* `data_`, `global_`, `loop_`, `save_`, `stop_`, 20, 22, 26–28, 31
case insensitivity, 23
Resolution limits, 100, 156
Resolution shells, 98, 100, 102, 155, 156, 160–161
RESTRAIN, 161
rhoCIF
atom properties, 141–143
category grouping, 89–91
classification and use, 141–143
design considerations, 141
dictionary, 290–294
history, 8
Rietveld refinement, 100, 117, 121–123, 128–129, 558
quality, 128
RTF, 482, 507

Sample (pdCIF), 118
characterization, 118
colour, 118
orientation, 127
preparation, 118
Satellite reflections, aperiodic structures, 131, 134, 137, 139
`save_` (STAR File keyword), 15, 16, 22, 25, 26, 31, 32, 40, 46

Save frame, 14–16, 18, 25, 26, 32, 45, 489–495, 500
  data definition block (DDL2), 21, 54, 61, 79, 81, 82
  MIF, 45, 46
  nested, 15, 26
  reference, 15, 22, 27, 46, 48, 49, 489–491, 494
  structural template (MIF), 48, 49
  uniqueness, 26, 46
  use in CIF dictionary, 20
Sawtooth function, *see* special modulation functions
Scan, *see* diffraction scan
Scattering factor, 103, 106
SCFS, *see* data exchange standards
Schema, 9
Schoenflies symbols, 206, 207
Scientific notation, 23
Scope, 15, 16, 47, 79, 115, 490, 492–494, 541
Secondary structure (protein), 181–182
Self-defining Text Archive and Retrieval File, *see* STAR File
Semantics, 2, 53, 491, 494, 499
Sequence, macromolecular, 166, 181–182
Sequence alignment, 186
Sequence heterogeneity, 145, 147, 176
SGML, 9, 482, 483, 515, 562, 563
  use in publication, 563
SHELX76, 557
SHELX97, 558
SHELXL, 483
Single pulse-counting detectors, 126
Small-molecule structures, described by coreCIF, 92
SMD, *see* data exchange standards
SMILES, *see* data exchange standards
Software, *see also* computer programs; data exchange standards; networking; protocols
  file transformation, 509–515
  performance, 481, 484
  rapid development tools, 517–522
  STAR File utilities, 488–498
  syntactic utilities for CIF, 499–525
Software design considerations, 481–487, 535–538
Space groups, 94, 206–208
  Hall symbol, 206, 207
  Hermann–Mauguin symbol, 206, 207
  origin choice, 110
  Schoenflies symbol, 206, 207
Space-group information
  coreCIF, 110–111
  symCIF, 206–208
Special characters, 22, 26, 27, 34–36, 107, 502, 507
Special modulation functions, 132, 136
Special positions, 103, 207
Specimen (pdCIF), 118
  characterization, 119
  preparation, 119
Spectroscopic data, 7
SPINE, 197
SQL, 523, 542
Standard Crystallographic File Structure (SCFS), *see* data exchange standards
Standard Generalized Markup Language, *see* SGML
Standard Molecular Data (SMD), *see* data exchange standards
Standard uncertainty
  attribute of data definition (DDL1), 34, 56, 59
  attribute of data definition (DDL2), 23, 64
  image detector gain, 200
  in geometric calculations, 109
  intensity measurements, 99, 126
  measurements of physical properties, 94
  overall measures (mmCIF), 159
  range constraint, 511, 529
  separate data names (mmCIF), 85, 147
Stanford Synchrotron Radiation Laboratory, 552
STAR File
  adoption for MIF, 44, 45
  basis for macromolecular structure ontology, 539
  binary tree representation, 494, 497
  conversion to HTML, 494, 496
STAR File
  data instances and context, 488
  development, 5
  format, 13
  grammar, 17–19
  lack of data typing, 20
  NMRStar, 9
  query language, 491–494, 496
  software utilities, 488–498
  specification, 13–19
  syntax, 5, 6, 14–19, 488, 495
  syntax restrictions in CIF, 23
  universal data language, 13, 53
Star_Base, 20, 484, 491–494, 496, 497, 499, 511
StarDDL, 14, 495
StarDOM, 497
starlib, 494, 497
StarMarkUp, 494, 495
STAR::Parser, 515–517
Star.vim, 494, 499
Stereochemistry, 47, 48, 50, 51
Stereogenic geometry (MIF), 50
`stop_` (STAR File keyword), 15, 16, 18, 22, 31, 40
String, *see* text string, quoted text string
Structural genomics, 197, 540, 542
Structural model
  coreCIF, 102–111
  mmCIF data items, 164–190
  msCIF, 134–136
  pdCIF, 123
Structural Proteomics in Europe (SPINE), 197
Structural template (MIF), 48, 49
Structure analysis
  coreCIF, 98–102
  mmCIF, 152–164
  msCIF, 134
  pdCIF, 121–123
Structure checking, 541
Structure factor, 99, 100, 558
Structure refinement, *see* refinement
Structure solution, 98, 106
Subcategory, 54, 62, 64, 67, 70, 79
Subgroup, 206, 208
Sublimation temperature, 107
Subscripts (markup in CIF), 36
Substructure query, 44, 48, 51, 108
Superscripts (markup in CIF), 36
Superspace, 131–133, 136, 137
Supplementary data, 3, 558
symCIF
  category grouping, 89–91
  classification and use, 206–208
  design considerations, 206
  dictionary, 459–466
  history, 8, 206
  relationship to coreCIF, 110
Symmetry, 8, 206–208
  coreCIF, 110–111
  mmcif, 190
  msCIF, 136
  superspace, 131, 132, 136, 137
  transformation, 109
Symmetry CIF (symmetry Crystallographic Information File), *see* symCIF
Symmetry operation, 206, 207
Symmetry reinforcement factor ε, 101
Symmetry-equivalent positions, 110, 111
Symmetry-equivalent reflections, 101
Synchrotron, 8, 117, 128, 199, 204
Synchrotron research, 8, 126
Syntax, 2, 3, 13, 14, 53, 499
  CBF, 39, 40
  CIF, 5, 6, 40, 53
  DDL, 55
  DDL1, 53
  MIF, 45
  STAR File, 5, 6, 14–20, 45
Syntax checker, 499–501, 507
Systematic absences, 94, 97, 100, 163

Tag, *see* data name
Taxonomy, 32
Tcl/Tk, 494, 495, 504, 515, 519
Temperature, *see* ambient conditions
Template, *see* structural template (MIF)
TEX, 482, 512, 513, 515, 562, 563
  use in publication, 114, 560
Text block, 17
Text field, *see* quoted text string
Text string, 14, 15, 17, 22, 24, 26, 33, 45, 58
Texture, 127
Time-of-flight neutron experiments, 8, 117, 127
Tokenization, 17, 21, 22, 24, 29, 30
Topology (chemical structure), 45
Topology (electron density), 141
Torsion angle, 110, 171, 173, 184
Typographic markup, 24, 34–36, 114, 563

Unicode, 22, 27
UNIMOL, 560
Unit
  attribute of data definition (DDL1), 56, 58, 59, 78
  attribute of data definition (DDL2), 64, 68, 83
Unit cell
  composite structures, 133
  contents, 107
  parameters and measurement conditions, 93, 148
  volume, 93
Units, 33, 54, 83, 563
  alternative (deprecated), 33, 79, 87
  conversion table (DDL2), 64, 68, 83
Universal data language, 13, 44, 53

Validation, 7, 29, 33, 48, 53, 54, 58, 61, 63, 74, 85, 87–89, 97, 102, 109, 491, 494, 496, 497, 499, 502, 504, 506, 507, 511, 526–528, 541, 543, 560
Validation reply form, 561
vcif, 499–501
Version information
  CBF, 38, 40
  data dictionary, 55, 60
vim, 494, 499
Virtual dictionary, *see* dictionary, composite
Visual Basic, 507, 560
Visualization, 145, 501, 503, 525, 557, 562
Voxel characteristics, 200–201

Wave vector, 131–133, 135, 137
Wavelength
  components, 95
  notation, 94
Web sites
  Cambridge Crystallographic Data Centre: http://www.ccdc.cam.ac.uk, 483
  CCP4: http://www.ccp4.ac.uk, 484
  Chemical Markup Language: http://cml.sf.net, 496
  Glasgow University, CIF software: http://www.chem.gla.ac.uk, 483
  GNU software: http://www.gnu.org, 486
  IUCr checkCIF service: http://checkcif.iucr.org, 561
  IUCr CIF dictionaries: ftp://ftp.iucr.org/pub/cifdics, 86
  IUCr CIF information: http://www.iucr.org/iucr-top/cif, 2, 7, 74, 75, 137, 525
  IUCr ftp site: ftp://ftp.iucr.org, 483
  IUCr journals: http://journals.iucr.org, 128, 560, 561, 566
  NASA data systems standards: http://ssdoo.gsfc.nasa.gov/nost, 483
  NeXus format: http://www.nexus.anl.gov/, 483
  Object Management Group: http://www.omg.org/, 541
  OpenMMS software: http://openmms.sdsc.edu, 541
  PDB deposition: http://deposit.pdb.org/, 197, 523

Web sites
PDB mmCIF site: http://mmcif.pdb.org/, 195, 197, 539, 540
PDB Protein Expression Purification and Crystallization Database: http://pepcdb.pdb.org/, 197
PDB software tools: http://sw-tools.pdb.org/, 197, 483, 484, 540–543
Research Collaboratory for Structural Bioinformatics: http://www.rcsb.org, 483
Research Collaboratory for Structural Bioinformatics data standardization: ftp://beta.rcsb.org/, 541
San Diego Supercomputer Center: http://www.sdsc.edu, 484
SMILES format: http://www.daylight.com/smiles/, 51
Structural Proteomics in Europe: http://www.spineurope.org/, 197
Web sites
World Wide Web Consortium: http://www.w3.org/, 30
White space, 14, 16–18, 21, 22, 24, 26, 27, 29, 31, 32, 38, 40, 538
elision of trailing, 22
Wilson *B* factor, 163
Working Party on Crystallographic Information, 5, 45
World Wide Web, 2, *see also* web sites, 5, 557
Worldwide PDB, 195
Wyckoff letter, 206, 208

X-Binary-Element-Type (MIME header), 39, 41
X-Binary-Size (MIME header), 39, 41
XD, 143
XML, 9, 30, 73, 481, 483, 496, 497, 515, 523, 526, 563, 564
conversion from CIF, 563, 564
XML
conversion from mmCIF, 9, 523, 564
mmCIF schema, 523, 541, 542
use in publication, 114, 563
XRAY, 3
X-ray film, 117, 126, 127
Xtal, 560

yacc, 486, 520
Yapps2, 517

Z (number of formula units in unit cell), 93, 107
zb, 519
Zeitschrift für Kristallographie, 557
ZINC (isomorph of CIF), 483, 518–519
zincCif, 519
zincGrep, 519
zincNl, 519
zincSubset, 519

Index of categories and data names

This is an index of data categories and data names referred to in this volume. The main headings (in capital letters) are categories; subordinate entries (in lower-case letters) refer to data items belonging to the parent category. Where the inital part of a data name can be constructed from the name of the category (which is normally the case), this is indicated by a long dash. Thus the first entry under the ARRAY_DATA heading refers to the data name **_array_data.array_id** *etc.* Primary definitions in the data dictionaries are indicated by **bold** page numbers. Data items appearing in examples are indicated by *italic* page numbers. Items in the Dictionary Definition Language (DDL1 and DDL2) and Molecular Information File (MIF) dictionaries appear twice in the index: alphabetically and under their own main headings.

ARRAY_DATA, 38, 39, 42, 199, **444**, 445, 446
—.array_id, *38, 39*, *42*, 199, **444**, *444*, *455, 456*, *554*
—.binary_id, *38, 39*, *42*, 43, 199, 203, **444**, *444*, 445, 449, 451, *455, 456*, *554*
—.data, 37, *38*, 39, *39*, 41, *42*, 199, **444**, *444*, *455, 456*, *554*
ARRAY_ELEMENT_SIZE, 199, 200, **445**
—.array_id, *39*, 200, **445**, *455, 456*
—.index, *39*, 200, **445**, *455, 456*
—.size, *39*, 200, **445**, *455, 456*
ARRAY_INTENSITIES, 199, 200, **445**, 550, 552
—.array_id, *38, 39*, 200, **445**, *445*, *455, 456*, *554*
—.binary_id, *38, 39*, 200, **445**, *455, 456*, *554*
—.gain, 200, **445**, *445*, *455, 456*, 550, *554*
—.gain_esd, 200, **446**, *455, 456*, 550, *554*
—.linearity, *38, 39*, 200, 445, *445*, **446**, *455, 456*, *554*
—.offset, 200, **446**
—.overload, *38, 39*, 200, *445*, **446**, *455, 456*, 551, *554*
—.scaling, 200, **446**
—.undefined_value, *38, 39*, 200, *445*, **446**, *455, 456*, *554*
ARRAY_STRUCTURE, 199, 200, 444, 445, **446**, 447
—.byte_order, *39*, 200, **446**, *446*, *455, 456*, *554*
—.compression_type, *39*, 200, **446**, *446*, *455, 456*, *554*
—.encoding_type, 39, *39*, 41, 200, 445, **446**, *446*, *455, 456*, *554*
—.id, *39*, 199, 200, 444, 445, **446**, *446*, 447, 449, 451, *455, 456*, *554*
ARRAY_STRUCTURE_LIST, 199–201, 203, 445, **447**, 450, 551, 552
—.array_id, *39*, 200, **447**, *455, 456*, *554*
—.axis_set_id, 200, 201, **447**, *455, 456*, *554*
—.dimension, *39*, 200, **447**, *455, 456*, *554*
—.direction, *39*, 200, **447**, *455, 456*, *554*
—.index, *39*, 200, 445, **447**, *455, 456*, *554*
—.precedence, *39*, 200, **447**, *455, 456*, *554*
ARRAY_STRUCTURE_LIST_AXIS, 199–201, 203, **447**, 450, 552
—.angle, 200, 201, **447**, *456*
—.angle_increment, 200, 201, **447**
—.angular_pitch, 200, 201, **447**, *456*
—.axis_id, 200, 201, **447**, *455, 456*, *554*
—.axis_set_id, 200, 201, **447**, *455, 456*, *554*
—.displacement, 200, 201, **447**, *455, 456*, *554*
—.displacement_increment, 200, 201, **448**, *455*, *554*
—.radial_pitch, 200, 201, **448**, *456*
_atom_attach_all, *see* MIF
_atom_attach_h, *see* MIF
_atom_attach_nh, *see* MIF
_atom_attach_ring, *see* MIF
_atom_charge, *see* MIF
_atom_cip, *see* MIF
_atom_coord_x, *see* MIF
_atom_coord_y, *see* MIF
_atom_coord_z, *see* MIF
_atom_environment, *see* MIF
_atom_frag_id, *see* MIF
_atom_frag_key, *see* MIF
_atom_id, *see* MIF
_atom_label, *see* MIF
_atom_mass_number, *see* MIF
_atom_radical_count, *see* MIF
_atom_spin_multiplicity, *see* MIF
_atom_type, *see* MIF
_atom_valency, *see* MIF
ATOM_LOCAL_AXES, 141, 142, **290**, 292, 293
—_atom[n] ($n = 0, 1, 2$), 141, *142*, **290**, *290*
—_atom_label, 141, 142, *142*, **290**, *290*, 291
—_ax[n] ($n = 1, 2$), 141, *142*, *290*, **291**
ATOM_RHO_MULTIPOLE, 141, 142, **291**
—_atom_label, 142, *143*, *291*, **292**, 293, 294
—_coeff_Pmn ($m = 0, \ldots, 4, n = -m, \ldots, m$), 142, *143*, *291*, **292**
—_coeff_Pc, 142, **292**, 293
—_coeff_Pv, 142, *143*, *291*, **292**
—_configuration, 142, *143*, **292**
—_core_source, 142, *143*, **292**
—_kappa, 142, 143, *143*, *291*, **292**, 293
—_kappa_prime$n$ ($n = 0, \ldots, 4$), 142, *143*, *291*, **292**
—_radial_function_type, 142, **293**
—_radial_slater_nn ($n = 0, \ldots, 4$), 142, *143*, **293**
—_radial_slater_zeta$n$ ($n = 0, \ldots, 4$), 142, *143*, **293**
—_scat_core, 142, 292, **293**
—_scat_valence, 142, 292, **293**, 294
—_valence_source, 142, *143*, **294**
ATOM_SITE, 56, *57*, 78, 90, 102–105, 109, 134, 135, 142, 146, 156, 164–167, 169, 171, 174, 175, 180, 182, 196, **210**, 214, 248, 250, 252, **270**, 288, 292, **294**, **295**, 296–299, 302–310, 312, 322, 323, 333, 352, 354, 362–371, 407, 410, 414, 420–426, 428–431, 436–441, 560
—.adp_type, 164, **296**
—_adp_type, 102, 103, *103*, **210**, *210*, 565
—.aniso_B[m][n] ($m, n = 1, 2, 3$), 164, 165, **296**, **297**
—.aniso_B[m][n]_esd ($m, n = 1, 2, 3$), **296, 297**
—_aniso_B_mn ($m, n = 1, 2, 3$), 102, *210*, **211**
—_aniso_label, 56, *57*, 78, *78*, 102–104, *104*, *210*, **211**, 506, 565
—.aniso_ratio, 165, **297**
—_aniso_ratio, 102, **211**
—_aniso_type_symbol, 102, *210*, **211**
—.aniso_U[m][n] ($m, n = 1, 2, 3$), 165, **298**, **299**
—.aniso_U[m][n]_esd ($m, n = 1, 2, 3$), **298, 299**
—_aniso_U_mn ($m, n = 1, 2, 3$), *78*, 102, *104*, **211**, 565
—.attached_hydrogens, 165, **299**, 333
—_attached_hydrogens, 88, 102, 105, **211**, 220
—.auth_asym_id, 165, **299**, 362, 364, 366, 367, 370, 421–423, 425, 430, 431, 438, 439
—.auth_atom_id, 165, **299**, 362–364, 366, 367, 370, 423, 436, 437, 440
—.auth_comp_id, 165, **300**, 363, 364, 366, 368, 370, 421–423, 425, 428, 430, 431, 438, 440
—.auth_seq_id, *5*, 165, 166, *166*, *295*, **300**, 363–366, 368, 370, 421–423, 425, 428, 430, 431, 436–438, 440
ATOM_SITE
—.B_equiv_geom_mean, 165, **300**
—_B_equiv_geom_mean, 102, **211**
—.B_equiv_geom_mean_esd, **300**
—.B_iso_or_equiv, *5*, *147*, 165, 166, *166*, *295*, **300**
—_B_iso_or_equiv, 102, **211**
—.B_iso_or_equiv_esd, **300**
—.calc_attached_atom, 165, **300**
—_calc_attached_atom, 102, *210*, **211**
—.calc_flag, 165, **301**
—_calc_flag, 102, 103, *103*, *210*, **211**
—.Cartn_x, *5*, *147*, 165, *166*, *295*, **301**
—_Cartn_x, 102, 103, **212**
—.Cartn_x_esd, **301**
—.Cartn_y, *5*, *147*, 165, *166*, *295*, **301**
—_Cartn_y, 102, 103, **212**
—.Cartn_y_esd, **301**
—.Cartn_z, *5*, *147*, 165, *166*, *295*, 301, **301**
—_Cartn_z, 102, 103, **212**
—.Cartn_z_esd, **301**
—.chemical_conn_number, 165, **301**, 333
—_chemical_conn_number, 102, 105, *210*, **212**
—.constraints, 165, **301**, 393, 394
—_constraints, 102, **212**, 244
—_description, 102, 138, **212**
—.details, 79, 165, **301**
—.disorder_assembly, 165, 168, **301**
—_disorder_assembly, 102, *103*, 105, *210*, **212**, 565
—.disorder_group, 165, 168, **301**
—_disorder_group, 102, *103*, 105, *210*, **212**, 565
—_displace_modulation_flag, 135, **270**
—.footnote_id, *147*, 165, 166, *166*, 167, *295*, **301**
—.fract_x, 33, 165, **301**
—_fract_x, *6*, *21*, 32, *78*, 90, 102, *103*, 105, 109, *124*, 129, 138, *210*, **212**, 236–238, *290*, 560, 565
—.fract_x_esd, **301**
—.fract_y, 165, **301**
—_fract_y, *6*, *21*, 33, *78*, 102, *103*, 105, 109, *124*, 129, 138, *210*, **212**, 236–238, *290*, 560, 565
—.fract_y_esd, **301**
—.fract_z, 165, **301**, 302
—_fract_z, *6*, *21*, 33, *78*, 102, *103*, 105, 109, *124*, 129, 138, *210*, **212**, 236–238, *290*, 560, 565
—.fract_z_esd, **302**
—.group_PDB, 165, 166, *166*, *295*, **302**
—.id, *5*, *147*, 165, 166, *166*, *295*, 300, **302**, 305, 363, 365, 366, 368, 370
—_label, *6*, *21*, 56, *57*, *75*, 78, *78*, 102, 103, *103*, 104, 105, 110, 129, 136, 138, 142, 165, *210*, 211, **212**, 213, 236–238, 272, 275–278, 280, 287, 290, *290*, 291, 292, **294**, 302, 560, 565
—.label_alt_id, *5*, 147, *147*, 165, 166, *166*, 169, *295*, **302**, 363, 365, 366, 368, 370, 371
—.label_asym_id, *5*, 147, *147*, 165, 166, *166*, *295*, 299, **302**, 363, 365, 366, 368, 371, 421–424, 426, 429–431, 440
—.label_atom_id, *5*, *147*, 165, 166, *166*, *295*, 299, **302**, 363, 365, 366, 368, 371, 436, 437

ATOM_SITE
—.label_comp_id, *5*, 147, *147*, 165, 166, *166*, 187, *295*, 300, **302**, 363, 365, 366, 368, 371, 421–424, 426, 429–431, 440
—._label_component_*n* ($n = 0, \ldots, 6$), 102, 103, **212**
—.label_entity_id, *5*, *147*, 165, 166, **302**
—.label_seq_id, *5*, *147*, 165, 166, *166*, *295*, 300, **302**, 363–365, 367–369, 371, 421–424, 426, 429–431, 436–438, 440
—._occ_modulation_flag, 135, **270**
—.occupancy, *5*, *147*, 165, *166*, *295*, **302**
—._occupancy, 103, *103*, 105, *124*, 129, 138, *210*, **212**, *290*, 565
—.occupancy_esd, **302**
—.refinement_flags, 165, 301, **302**, 303, 393, 394
—._refinement_flags, 103, 105, 212, **213**
—.refinement_flags_adp, 165, **302**
—._refinement_flags_adp, 103, 105, **213**
—.refinement_flags_occupancy, 165, **302**
—._refinement_flags_occupancy, 103, **213**
—.refinement_flags_posn, 165, 302, **303**
—._refinement_flags_posn, 103, *103*, 105, **213**
—.restraints, 165, **303**
—._restraints, 103, **213**
—._subsystem_code, 135, **270**
—.symmetry_multiplicity, 165, **303**
—._symmetry_multiplicity, 103, *124*, **213**
—.thermal_displace_type, 165, **303**
—._thermal_displace_type, *78*, 103, *124*, **213**
—.type_symbol, *5*, 84, *147*, 165, *166*, *295*, **303**
—._type_symbol, *21*, 84, 103, *103*, 104, *124*, 212, **213**, 560
—.U_equiv_geom_mean, 165, **303**
—._U_equiv_geom_mean, 103, **213**
—.U_equiv_geom_mean_esd, **303**
—.U_iso_or_equiv, 165, **303**
—._U_iso_or_equiv, *21*, 78, *78*, 103, *103*, 105, *124*, 129, *210*, **213**, 560, 565
—.U_iso_or_equiv_esd, **303**
—._U_modulation_flag, 135, **270**
—.Wyckoff_symbol, 165, **303**
—._Wyckoff_symbol, 103, **213**, 560
ATOM_SITE_ANISOTROP, 102, 164–166, 295–299, **303**, 304–307
—.B[*m*][*n*] ($m, n = 1, 2, 3$), 165, **304**, **305**
—.B[*m*][*n*]_esd ($m, n = 1, 2, 3$), **304, 305**
—.id, 165, 166, *303*, **305**
—.ratio, 165, **305**
—.type_symbol, 165, *303*, **305**
—.U[*m*][*n*] ($m, n = 1, 2, 3$), 165, *303*, **305**, **306**, **307**
—.U[*m*][*n*]_esd ($m, n = 1, 2, 3$), **306, 307**
ATOM_SITE_DISPLACE_FOURIER, 134–136, **270**, 272, 278, 281
—._atom_site_label, 135, 138, *270*, **271**, *271*, 272, *272, 273*, 274, *274*, 275, 278
—._axis, 135, 138, *270, 271*, **272**, *272, 273*, 274, *274*, 275
—._id, 135, 136, 138, *270, 271*, **272**, *272, 273*, 274, *274*
—._wave_vector_seq_id, 135, 138, *270, 271*, **272**, *272–274*
ATOM_SITE_DISPLACE_FOURIER_PARAM, 134–136, 270, **272**, 279
—._cos, 135, 138, *272, 273*, **274**, *274*
—._id, 135, 138, *272, 273*, **274**, *274*
—._modulus, 135, **274**, *274*
—._phase, 135, **274**, *274*
—._sin, 135, 136, 138, *272–274*, **275**
ATOM_SITE_DISPLACE_SPECIAL_FUNC, 134, 136, **275**
—._atom_site_label, 136, **275**, *275*
—._sawtooth_ax, 136, **275**, *275*
—._sawtooth_ay, 136, **275**, *275*
—._sawtooth_az, 136, **275**, *275*
—._sawtooth_c, 136, **275**, *275*
—._sawtooth_w, 136, **275**, *275*
ATOM_SITE_FOURIER_WAVE_VECTOR, 134–136, **275**
—._description, 135, 138, *270–274*, **275**, *275*
—._seq_id, 135, 136, 138, *270, 271*, 272, *272–275*, **276**, 279, 280
—._x, 135, 138, *270–275*, **276**
—._y, 135, **276**
—._z, 135, *271*, *273, 274*, **276**
ATOM_SITE_OCC_FOURIER, 134–136, **276**
—._atom_site_label, 135, **276**, *276*, 277
—._id, 135, **276**, *276*
—._wave_vector_seq_id, 135, **276**, *276*, 277
ATOM_SITE_OCC_FOURIER_PARAM, 134, 135, **276**
—._cos, 135, **276**
—._id, 135, **276**, *276*
—._modulus, 135, *276*, **277**
—._phase, 135, *276*, **277**
—._sin, 135, **277**
ATOM_SITE_OCC_SPECIAL_FUNC, 134, 136, **277**
—._atom_site_label, 136, **277**, *277*
—._crenel_c, 136, **277**, *277*
—._crenel_w, 136, **277**, *277*
ATOM_SITE_PHASON, 134, 135, **278**
—._atom_site_label, 135, **278**
—._coeff, 135, **278**
—._formula, 135, **278**
ATOM_SITE_ROT_FOURIER, 134–136, 270, **278**, 279
—._atom_site_label, 135, 138, 272, **278**, *278*, 279, 280
—._axis, 135, 138, **278**, *278*, 279, 280
—._id, 135, 138, **278**, *278*, 279
—._wave_vector_seq_id, 135, 138, *278*, **279**
ATOM_SITE_ROT_FOURIER_PARAM, 134–136, 272, 278, **279**
—._cos, 135, 138, 274, **279**, *279*
—._id, 135, 138, **279**, *279*
—._modulus, 135, 274, **279**, *279*
—._phase, 135, 274, **279**, *279*
—._sin, 135, 138, 275, *279*, **280**
ATOM_SITE_U_FOURIER, 134–136, **280**
—._atom_site_label, 135, **280**, *280*, 281
—._id, 135, **280**, *280*, 281
—._tens_elem, 135, **280**, *280*, 281
—._wave_vector_seq_id, 135, **280**, *280*
ATOM_SITE_U_FOURIER_PARAM, 134, 135, **280**
—._cos, 135, **280**, 281
—._id, 135, *280*, **281**
—._modulus, 135, *280*, **281**
—._phase, 135, *280*, **281**
—._sin, 135, **281**
ATOM_SITES, 90, 102, 103, 105, 164, 167, **214**, **307**
—._Cartn_tran_matrix_*mn* ($m, n = 1, 2, 3$), 105, **214**, *214*
—._Cartn_tran_vector_*n* ($n = 1, 2, 3$), 105, **214**
—.Cartn_transf_matrix[*m*][*n*] ($m, n = 1, 2, 3$), 167, **307**, *307*, **308**
—.Cartn_transf_vector[*n*] ($n = 1, 2, 3$), 167, *307*, **308**
—.Cartn_transform_axes, 167, 301, 307, *307*, **308**, 309, 310, 382, 383
—._Cartn_transform_axes, 105, 212, **214**, *214*
—.entry_id, 167, *307*, **308**
—._fract_tran_matrix_*mn* ($m, n = 1, 2, 3$), 90, 105, 106, **214**
—._fract_tran_vector_*n* ($n = 1, 2, 3$), 106, **214**
—.fract_transf_matrix[*m*][*n*] ($m, n = 1, 2, 3$), 167, **309**
—.fract_transf_vector[*n*] ($n = 1, 2, 3$), 167, **310**
—.solution_hydrogens, 167, **310**
—._solution_hydrogens, 98, 106, **214**
—.solution_primary, 167, **310**
—._solution_primary, 98, 106, **214**
—.solution_secondary, 167, **310**
—._solution_secondary, 106, **214**
ATOM_SITES
—.special_details, 167, **310**
—._special_details, 106, **214**
ATOM_SITES_ALT, 147, 164, 166, 168, 302, **310**, 311, 312, 423, 424, 426, 429–431, 440
—.details, 168, *168*, *310*, **311**
—.id, 166, 168, *168*, 302, *310*, **311**, 312, 423, 424, 426, 429–431, 440
ATOM_SITES_ALT_ENS, 164, 168, **311**, 312
—.details, 168, *168*, **311**, *311*
—.id, 168, *168*, **311**, *311*, 312
ATOM_SITES_ALT_GEN, 164, 168, 169, **312**
—.alt_id, 168, *168*, **312**, *312*
—.ens_id, 168, *168*, **312**, *312*
ATOM_SITES_DISPLACE_FOURIER, 134–136, **281**
—._axes_description, 135, 272, 274, 275, **281**
ATOM_SITES_FOOTNOTE, 164, 166, 167, 301, **312**
—.id, 166, 167, *167*, 301, **312**, *312*
—.text, 167, *167*, **312**, *312*
ATOM_SITES_MODULATION, 134–136, **281**
—._global_phase_t_*n* ($n = 1, \ldots, 8$), 135, **281**
ATOM_SITES_ROT_FOURIER, 134–136, **282**
—._axes_description, 135, 278, **282**
ATOM_TYPE, 90, 102–104, 106, 164, 165, 167, **214**, 303, 305, **312**, 323, 333, 382
—._analytical_mass_%, 106, 118, **215**
—.analytical_mass_percent, 167, **313**
—.description, *5*, 167, **313**
—._description, 104, 106, **215**, 565
—.number_in_cell, *5*, 167, *312*, **313**
—._number_in_cell, 106, *124*, *214*, **215**
—.oxidation_number, 167, *312*, **313**
—._oxidation_number, 90, 106, *214*, **215**
—.radius_bond, 167, **313**
—._radius_bond, *6*, 106, **215**
—.radius_contact, 167, **313**
—._radius_contact, 106, **215**
—.scat_Cromer_Mann_a*n* ($n = 1, \ldots, 4$), 167, *312*, **313**
—._scat_Cromer_Mann_a*n* ($n = 1, \ldots, 4$), 106, **215**
—.scat_Cromer_Mann_b*n* ($n = 1, \ldots, 4$), 167, *312*, **313**
—._scat_Cromer_Mann_b*n* ($n = 1, \ldots, 4$), 106, **215**
—.scat_Cromer_Mann_c, 167, *312*, **313**
—._scat_Cromer_Mann_c, 106, **215**
—.scat_dispersion_imag, 167, *312*, **313**
—._scat_dispersion_imag, 106, *214*, **215**, 565
—.scat_dispersion_real, 167, *312*, **313**
—._scat_dispersion_real, 106, *214*, **215**, 565
—.scat_dispersion_source, 167, **314**
—._scat_dispersion_source, 106, **215**
—.scat_length_neutron, 167, **314**
—._scat_length_neutron, 106, *125*, **215**
—.scat_source, 167, *312*, **314**
—._scat_source, *6*, 106, *125*, *214*, **215**, 565
—.scat_versus_stol_list, 167, **314**
—._scat_versus_stol_list, 106, **215**
—.symbol, *5*, 167, 303, 305, *312*, **314**, 323, 333, 382
—._symbol, *6*, 104, 106, *124, 125*, 211–213, *214*, **215**, 220, 565
AUDIT, 114, 115, 132, 136, 137, 194, **215**, **314**
—._block_code, 115, *115*, 116, *116*, 137, *139*, 194, **215**, *215*, 217, *271*, *273*, 282
—.creation_date, *5*, 194, **314**, *314*
—._creation_date, *6*, 85, 115, *115*, **215**, *215*
—.creation_method, *5*, 194, **314**, *314*
—._creation_method, *6*, 115, *115*, **215**, *215*
—.revision_id, 194, **314**, *314*
—.update_record, *5*, 194, **314**, *314*
—._update_record, *6*, 115, *115*, **215**, *215*
AUDIT_AUTHOR, 114, 115, 194, **216**, **314**
—.address, 194, *314*, **315**
—._address, 115, **216**, *216*

AUDIT_AUTHOR
—.name, 194, *314*, **315**
—._name, 115, **216**, *216*
AUDIT_CONFORM, 25, 114, 115, 194, **216**, **315**
—.dict_location, *35*, 194, **315**, *315*
—._dict_location, *35*, 85, 115, **216**, *216*
—.dict_name, *35*, 85, 87, 194, **315**, *315*
—._dict_name, *35*, 85, 87, 115, **216**, *216*
—.dict_version, *35*, 194, **315**, *315*
—._dict_version, *35*, 85, 115, **216**, *216*
AUDIT_CONTACT_AUTHOR, 114, 115, 194, **216**, **315**
—.address, 194, **315**, *315*
—._address, 115, **216**, *216*
—.email, 194, **315**, *315*
—._email, 115, **216**, *216*
—.fax, 194, **315**, *315*
—._fax, 115, **216**, *216*
—.name, 194, **315**, *315*
—._name, 115, **216**, *216*
—.phone, 194, **315**, *315*
—._phone, 115, *216*, **217**
AUDIT_LINK, 25, 114, 115, 132, 136, 137, 194, 195, **217**, 270, **282**, **316**
—.block_code, 195, **316**, *316*
—._block_code, 115, 116, 137, 139, **217**, *217*, *271*, *273*, *282*
—.block_description, 195, **316**, *316*
—._block_description, 115, 116, 139, **217**, *217*, *271*, *273*, *282*
AXIS, 199, 201, 202, 204, 205, 447, **448**, 449, 450, 452, 453, 457, 458, 551, 552
—.depends_on, 202, **448**, *448*, *455, 456*, *554*
—.equipment, 202, **448**, *448*, *455, 456*, *554*
—.id, 201–203, 205, 447, 448, *448*, **449**, 450, 452, *455, 456*, 457, 458, *554*
—.offset[*n*] ($n = 1, 2, 3$), 202, *448*, **449**, *455, 456*, *554*
—.type, 202, *448*, **449**, *455, 456*, *554*
—.vector[*n*] ($n = 1, 2, 3$), 202, *448*, **449**, *455, 456*, *554*

_bond_cip, *see* MIF
_bond_environment, *see* MIF
_bond_id_1, *see* MIF
_bond_id_2, *see* MIF
_bond_type_casreg3, *see* MIF
_bond_type_ccdc, *see* MIF
_bond_type_mif, *see* MIF

_category, *see* DDL1
_category.description, *see* DDL2
_category.id, *see* DDL2
_category.implicit_key, *see* DDL2
_category.mandatory_code, *see* DDL2
_category_examples.case, *see* DDL2
_category_examples.detail, *see* DDL2
_category_examples.id, *see* DDL2
_category_group.category_id, *see* DDL2
_category_group.id, *see* DDL2
_category_group_list.description, *see* DDL2
_category_group_list.id, *see* DDL2
_category_group_list.parent_id, *see* DDL2
_category_key.id, *see* DDL2
_category_key.name, *see* DDL2
CATEGORY, 64, 68, 69, **473**
CATEGORY_EXAMPLES, 64, 69, **473**
CATEGORY_GROUP, 64, 69, **473**
CATEGORY_GROUP_LIST, 64, 69, **473**
CATEGORY_KEY, 64, 69, **473**
CATEGORY_METHODS, 69, **473**
CELL, 93, 101, 132, 133, 148, 149, 157, **217**, 236–238, 249, **282**, 286, 287, **316**, 356, 362, 364, 366, 367, 369, 409, 410
—.angle_alpha, *5*, 148, *149*, **316**, *316*, 318
—._angle_alpha, *6*, *21*, 93, *124*, **217**, *217*, 218, *559*, *563*, 565
—.angle_alpha_esd, **316**
—.angle_beta, *5*, 148, *149*, **316**, *316*, 318
—._angle_beta, *6*, *21*, 93, *124*, **217**, *217*, 218, *559*, *563*, 565
—.angle_beta_esd, **316**
—.angle_gamma, *5*, 148, *149*, **316**, *316*, 318
—._angle_gamma, *6*, *21*, 93, *124*, **217**, *217*, 218, *559*, *563*, 565
—.angle_gamma_esd, **316**
—.details, 148, *149*, **316**, *316*
—.entry_id, 148, *149*, 195, **316**, *316*
—.formula_units_Z, 148, **316**
—._formula_units_Z, *6*, 93, 107, **217**, *563*, 565
—.length_a, *5*, 144, 148, *148, 149*, 301, *316*, **317**, 318, 383
—._length_a, *6*, *21*, 33, 93, *124*, 144, *148*, **217**, *217*, 218, *500*, *559*, *563*, 565
—.length_a_esd, *148, 149*, *316*, **317**
—.length_b, *5*, 148, *149*, 301, *316*, **317**, 318, 383
—._length_b, *6*, *21*, 93, *124*, **217**, *217*, 218, *500*, *559*, *563*, 565
—.length_b_esd, *149*, *316*, **317**
—.length_c, *5*, 148, *149*, 301, *316*, **317**, 318, 383
—._length_c, *6*, *21*, 93, *124*, **217**, *217*, *559*, *563*, 565
—.length_c_esd, *149*, *316*, **317**
—._measurement_pressure, 93, **217**
—._measurement_radiation, 93, **217**
—._measurement_reflns_used, *6*, 93, **217**, *217*, *563*, 565
—._measurement_temperature, *6*, 93, **217**, *217*, *563*, 565
—._measurement_theta_max, *6*, 93, **217**, *217*, *563*, 565
—._measurement_theta_min, *6*, 93, **217**, *217*, *563*, 565
—._measurement_wavelength, 93, **217**
—._modulation_dimension, 133, *138*, 276, **282**, 283–289
—.reciprocal_angle_alpha, 148, **317**
—._reciprocal_angle_alpha, 93, **218**
—.reciprocal_angle_alpha_esd, **317**
—.reciprocal_angle_beta, 148, **317**
—._reciprocal_angle_beta, 93, **218**
—.reciprocal_angle_beta_esd, **317**
—.reciprocal_angle_gamma, 148, **317**
—._reciprocal_angle_gamma, 93, **218**
—.reciprocal_angle_gamma_esd, **317**
—._reciprocal_basis_description, 133, *137, 138*, **282**, 284, 285, 288
—.reciprocal_length_a, 148, **317**
—._reciprocal_length_a, 93, **218**
—.reciprocal_length_a_esd, **317**
—.reciprocal_length_b, 148, **318**
—._reciprocal_length_b, 93, **218**
—.reciprocal_length_b_esd, **318**
—.reciprocal_length_c, 148, **318**
—._reciprocal_length_c, 93, **218**
—.reciprocal_length_c_esd, **318**
—._special_details, 93, **218**
—.volume, 148, *149*, *316*, **318**
—._volume, *6*, 93, *124*, *217*, **218**, *563*, 565, 566
—.volume_esd, *316*, **318**
—.Z_PDB, 148, **318**
CELL_MEASUREMENT, 148, 149, **318**
—.entry_id, 148, *149*, **318**, *318*
—.pressure, 148, **318**
—.pressure_esd, **318**
—.radiation, 148, **318**
—.reflns_used, 148, **318**, *318*
—.temp, 148, *149*, **318**, *318*
—.temp_esd, *149*, **318**, *318*
—.theta_max, 148, *149*, *318*, **319**
—.theta_min, 148, *149*, *318*, **319**
—.wavelength, 148, *149*, 318, *318*, **319**
CELL_MEASUREMENT_REFLN, 93, 148, **218**, 318, **319**
—.index_h, 148, **319**, *319*
—._index_h, 93, **218**
—.index_k, 148, **319**, *319*
—._index_k, 93, **218**
—.index_l, 148, **319**, *319*
—._index_l, 93, **218**
—.theta, 148, **319**, *319*
—._theta, 93, **218**
CELL_SUBSYSTEM, 132, 133, **282**
—._code, 133, 135, 138, 270, *270*, *272*, **282**, *282*
—._description, 133, 138, *270*, *272*, *282*, **283**
—._matrix_W_*m*_*n* ($m, n = 1, \ldots, 11$), 85, 133, 138, *270*, *272*, *282*, **283**
CELL_SUBSYSTEMS, 132, 133, **284**
—._number, 133, *138*, *282*, 283, **284**, *284*
CELL_WAVE_VECTOR, 133, **284**
—._seq_id, 133, *137*, **284**, *284*, 285, 286, 288
—._x, 133, *137*, **284**, *284*
—._y, 133, **284**
—._z, 133, **284**
CELL_WAVE_VECTORS, 133, **284**
—._meas_details, 133, **284**, *284*
—._pressure_max, 133, **285**
—._pressure_min, 133, **285**
—._temp_max, 133, **285**
—._temp_min, 133, **285**
—._variation, 133, **285**
CHEM_COMP, 146, 164, 166, 169–173, 175, 302, **319**, 320–325, 353, 355, 434, 438
—.formula, *147*, 169, *172*, **319**
—.formula_weight, 169, *172*, **319**
—.id, *147*, 166, 169, *172*, 302, **319**, *319*, 320–325, 355, 434, 438
—.model_details, 169, 171, **320**
—.model_erf, 169–171, **320**
—.model_source, 170, 171, *319*, **320**
—.mon_nstd_class, 170, **320**
—.mon_nstd_details, 170, **320**
—.mon_nstd_flag, *147*, 170, **320**
—.mon_nstd_parent, 170, **320**
—.mon_nstd_parent_comp_id, 170, **320**
—.name, *147*, 170, *172*, *319*, **320**
—.number_atoms_all, 170, *172*, **320**
—.number_atoms_nh, 170, *172*, **320**
—.one_letter_code, 170, 171, **320**
—.three_letter_code, 170, 171, **320**
—.type, 170–172, *172*, **321**, 324, 325
CHEM_COMP_ANGLE, 169–171, 319, **321**
—.atom_id_1, 170, **321**, *321*, 327
—.atom_id_2, 170, **321**, *321*
—.atom_id_3, 170, **321**, *321*, 327
—.comp_id, 170, **321**, *321*
—.value_angle, 170, **321**, *321*
—.value_angle_esd, **321**
—.value_dist, 170, **321**, *321*, 327
—.value_dist_esd, **321**
CHEM_COMP_ATOM, 166, 169–172, 302, 319, 321, **322**, 323–326, 423, 424, 440
—.alt_atom_id, 170, **322**
—.atom_id, 166, 170, *172*, 302, 321, **322**, *322*, 323–325, 423, 424, 440
—.charge, 170, **322**
—.comp_id, 170, *172*, **322**, *322*, 326
—.model_Cartn_x, 170, *172*, **322**, *322*
—.model_Cartn_x_esd, **322**
—.model_Cartn_y, 170, *172*, **322**, *322*
—.model_Cartn_y_esd, **322**
—.model_Cartn_z, 170, *172*, *322*, **323**
—.model_Cartn_z_esd, **323**
—.partial_charge, 170, **323**
—.substruct_code, 170, *322*, **323**
—.type_symbol, 170, *172*, *322*, **323**
CHEM_COMP_BOND, 169–172, 182, 319, **323**
—.atom_id_1, 170, *172*, **323**, *323*
—.atom_id_2, 170, *172*, **323**, *323*
—.comp_id, 170, *172*, **323**, *323*
—.value_dist, 170, *172*, **323**
—.value_dist_esd, *172*, **323**
—.value_order, 170, *172*, **323**, *323*

CHEM_COMP_CHIR, 169–171, 173, **323**, 324
—.atom_config, 170, **324**
—.atom_id, 170, *323*, **324**
—.comp_id, 170, *323*, **324**
—.id, 170, *323*, **324**
—.number_atoms_all, 170, **324**
—.number_atoms_nh, 170, **324**
—.volume_flag, 170, **324**
—.volume_three, 170, **324**
—.volume_three_esd, **324**
CHEM_COMP_CHIR_ATOM, 169–171, 173, 323, **324**
—.atom_id, 170, **324**, *324*
—.chir_id, 170, **324**, *324*
—.comp_id, 170, **324**, *324*
—.dev, 170, **324**
CHEM_COMP_LINK, 169, 170, 172, **324**
—.details, 170, **324**
—.link_id, 170, 172, **324**
—.type_comp_1, 170, **324**, 327
—.type_comp_2, 170, **325**, 327
CHEM_COMP_PLANE, 169–171, 173, **325**
—.comp_id, 170, **325**, *325*
—.id, 170, **325**, *325*
—.number_atoms_all, 170, **325**
—.number_atoms_nh, 170, **325**
CHEM_COMP_PLANE_ATOM, 169–171, 173, **325**
—.atom_id, 170, **325**, *325*
—.comp_id, 170, **325**, *325*
—.dist_esd, **325**
—.plane_id, 170, **325**, *325*
CHEM_COMP_TOR, 169–172, **325**, 326
—.atom_id_1, 170, *172*, **325**, *325*, 326
—.atom_id_2, 170, *172*, **325**, *325*
—.atom_id_3, 170, *172*, **325**, *325*
—.atom_id_4, 170, *172*, **325**, *325*, 326
—.comp_id, 170, *172*, **325**, *325*
—.id, 170, *172*, *325*, **326**
CHEM_COMP_TOR_VALUE, 169, 170, 172, 325, **326**
—.angle, 170, *172*, **326**, *326*
—.angle_esd, **326**
—.comp_id, 170, *172*, **326**, *326*
—.dist, 170, *172*, **326**, *326*
—.dist_esd, **326**
—.tor_id, 170, *172*, **326**, *326*
CHEM_LINK, 164, 169, 172–174, 182, 324, **326**, 327–330, 353
—.details, 172, **326**
—.id, 172, 174, 324, **326**, 327–330, 353
CHEM_LINK_ANGLE, 169, 172, 173, **326**
—.atom_1_comp_id, 172, *173*, *326*, **327**
—.atom_2_comp_id, 172, *173*, *326*, **327**
—.atom_3_comp_id, 172, *173*, *326*, **327**
—.atom_id_1, 172, *173*, *326*, **327**
—.atom_id_2, 172, *173*, *326*, **327**
—.atom_id_3, 172, *173*, *326*, **327**
—.link_id, 172, *173*, *326*, **327**
—.value_angle, 172, *173*, *326*, **327**
—.value_angle_esd, *173*, *326*, **327**
—.value_dist, 172, **327**
—.value_dist_esd, **327**
CHEM_LINK_BOND, 169, 172, 173, **327**
—.atom_1_comp_id, 172, *173*, **327**, *327*
—.atom_2_comp_id, 172, *173*, **327**, *327*
—.atom_id_1, 172, *173*, *327*, **328**
—.atom_id_2, 172, *173*, *327*, **328**
—.link_id, 172, *173*, *327*, **328**
—.value_dist, 172, *173*, *327*, **328**
—.value_dist_esd, *173*, *327*, **328**
—.value_order, 172, **328**
CHEM_LINK_CHIR, 169, 172, 173, **328**, 329
—.atom_comp_id, 172, **328**
—.atom_config, 172, **328**
—.atom_id, 172, **328**
—.id, 172, **328**, 329
—.link_id, 172, **328**
CHEM_LINK_CHIR
—.number_atoms_all, 172, **328**
—.number_atoms_nh, 172, **328**
—.volume_flag, 172, **328**
—.volume_three, 172, **328**
—.volume_three_esd, **328**
CHEM_LINK_CHIR_ATOM, 169, 172, 173, 328, **329**
—.atom_comp_id, 172, **329**
—.atom_id, 172, **329**
—.chir_id, 172, **329**
—.dev, 172, **329**
CHEM_LINK_PLANE, 169, 172, 173, **329**
—.id, 172, **329**
—.link_id, 172, **329**
—.number_atoms_all, 172, **329**
—.number_atoms_nh, 172, **329**
CHEM_LINK_PLANE_ATOM, 169, 172, 173, **329**
—.atom_comp_id, 172, **329**
—.atom_id, 172, **329**
—.plane_id, 172, **329**
CHEM_LINK_TOR, 169, 172, 173, **329**, 330
—.atom_1_comp_id, 172, **329**
—.atom_2_comp_id, 172, **329**
—.atom_3_comp_id, 172, **329**
—.atom_4_comp_id, 172, **330**
—.atom_id_1, 172, **330**
—.atom_id_2, 172, **330**
—.atom_id_3, 172, **330**
—.atom_id_4, 172, **330**
—.id, 172, **330**
—.link_id, 172, **330**
CHEM_LINK_TOR_VALUE, 169, 172, 174, 329, **330**
—.angle, 172, **330**
—.angle_esd, **330**
—.dist, 172, **330**
—.dist_esd, **330**
—.tor_id, 172, **330**
CHEMICAL, 105–107, 164, 169, 174, **218**, **330**, 352
—.absolute_configuration, 169, **330**
—_absolute_configuration, 106, **218**, 567
—.compound_source, 169, **331**
—_compound_source, *6*, 106, **218**
—.entry_id, *38*, 169, *330*, **331**
—.melting_point, 169, **331**
—_melting_point, 106, 118, **218**, 219, 565
—.melting_point_gt, 169, **331**
—_melting_point_gt, 106, **219**
—.melting_point_lt, 169, **331**
—_melting_point_lt, 106, **219**
—.name_common, *38*, 169, **331**
—_name_common, 106, **219**
—.name_mineral, 169, **331**
—_name_mineral, 106, **219**
—.name_structure_type, 169, **331**
—_name_structure_type, 106, **219**
—.name_systematic, 169, *330*, **331**
—_name_systematic, *6*, *21*, *35*, 106, *218*, **219**
—.optical_rotation, 169, **331**
—_optical_rotation, 107, **219**
—.properties_biological, 169, **331**
—_properties_biological, 107, **219**
—.properties_physical, 169, **331**
—_properties_physical, 107, **219**
—.temperature_decomposition, 169, **331**
—_temperature_decomposition, 107, **219**
—.temperature_decomposition_esd, **331**
—.temperature_decomposition_gt, 169, **332**
—_temperature_decomposition_gt, 107, **219**
—.temperature_decomposition_lt, 169, **332**
—_temperature_decomposition_lt, 107, **219**
—.temperature_sublimation, 169, **332**
—_temperature_sublimation, 107, **219**
—.temperature_sublimation_esd, **332**
—.temperature_sublimation_gt, 169, **332**
—_temperature_sublimation_gt, 107, **219**
CHEMICAL
—.temperature_sublimation_lt, 169, **332**
—_temperature_sublimation_lt, 107, **219**
CHEMICAL_CONN_ATOM, 105–108, 169, **219**, 301, **332**, 333
—.charge, 169, **332**
—_charge, 107, **220**
—.display_x, 169, **332**, *332*
—_display_x, 105, 107, *108*, **220**, *220*
—.display_y, 169, **332**, *332*
—_display_y, 105, 107, *108*, **220**, *220*
—.NCA, 169, *332*, **333**
—_NCA, 105, 107, *108*, **220**, *220*
—.NH, 169, *332*, **333**
—_NH, 105, 107, *108*, **220**, *220*
—.number, 169, 301, *332*, **333**
—_number, 105, 107, *108*, 212, **220**, *220*
—.type_symbol, 169, *332*, **333**
—_type_symbol, 105, 107, *108*, **220**, *220*
CHEMICAL_CONN_BOND, 105–108, 169, **220**, 332, **333**
—.atom_1, 169, **333**, *333*
—_atom_1, 107, *108*, **220**, *220*
—.atom_2, 169, **333**, *333*
—_atom_2, 105, 107, *108*, **220**, *220*
—.type, 169, 171, **333**, *333*
—_type, 107, *108*, **220**, *220*
CHEMICAL_FORMULA, 106, 107, 169, **221**, **333**, 334
—.analytical, 169, 333, **334**
—_analytical, 107, *107*, **221**
—.entry_id, 169, *333*, **334**
—.iupac, 169, **334**
—_iupac, 107, *108*, **221**, *221*
—.moiety, 169, 316, 333, *333*, **334**
—_moiety, *6*, *21*, *35*, 85, 107, *108*, **221**, *221*, *559*, *563*, 565
—.structural, 169, 316, 333, **334**
—_structural, 107, *108*, **221**, *221*
—.sum, 169, 316, 333, *333*, **334**
—_sum, *6*, *35*, 107, *108*, **221**, *221*, 488, *559*, 565, 566
—.weight, 169, *333*, **334**
—_weight, *6*, *21*, *35*, 107, **221**, *221*, *559*, 565, 566
—.weight_meas, 169, **334**
—_weight_meas, 107, **221**
CITATION, 83, 112, 190, 191, **222**, **334**, 335–337, 416
—.abstract, 190, **335**
—_abstract, 112, **222**
—.abstract_id_CAS, 190, **335**
—_abstract_id_CAS, 112, **222**
—.book_id_ISBN, 190, *334*, **335**
—_book_id_ISBN, 112, **222**, *222*
—.book_publisher, 190, *334*, **335**
—_book_publisher, 112, **222**, *222*
—.book_publisher_city, 190, **335**
—_book_publisher_city, 112, **222**
—.book_title, 190, *334*, **335**
—_book_title, 112, **222**, *222*
—.coordinate_linkage, 190, *334*, **335**
—_coordinate_linkage, 112, **222**, *222*
—.country, 190, *334*, **335**
—_country, 112, **222**, *222*
—.database_id_CSD, 190, **335**
—_database_id_CSD, 112, **222**
—.database_id_Medline, 190, **335**
—_database_id_Medline, 112, **222**
—.details, 191, *334*, **335**
—.id, *5*, 65, 190, 191, *334*, **335**, 336, 337, 416
—_id, 112, *112*, **222**, *222*, 223, 224
—.journal_abbrev, *5*, 190, *334*, **335**
—_journal_abbrev, 112, *112*, **222**, *222*
—.journal_full, 190, **335**
—_journal_full, 112, **223**
—.journal_id_ASTM, 190, *334*, **335**
—_journal_id_ASTM, *5*, 112, *222*, **223**
—.journal_id_CSD, 190, *334*, **336**

CITATION
—._journal_id_CSD, 112, **223**
—.journal_id_ISSN, 190, *334*, **336**
—._journal_id_ISSN, *5*, 112, *112*, *222*, **223**
—.journal_issue, 190, *334*, **336**
—._journal_issue, 112, *222*, **223**
—.journal_volume, *5*, 190, *334*, **336**
—._journal_volume, 112, *112*, *222*, **223**
—.language, 190, **336**
—._language, 112, **223**
—.page_first, *5*, 190, *334*, **336**
—._page_first, 112, *112*, *222*, **223**
—.page_last, 191, *334*, **336**
—._page_last, 112, *112*, *222*, **223**
—._special_details, 112, *222*, **223**
—.title, 191, *334*, **336**
—._title, *5*, 112, *112*, *222*, **223**
—.year, *5*, 191, *334*, **336**
—._year, 112, *112*, *222*, **223**
CITATION_AUTHOR, 83, 112, 190, 191, **223**, **336**
—.citation_id, *5*, 191, **336**, *336*
—._citation_id, 112, *112*, **223**, *223*
—.name, *5*, 191, **336**, *336*
—._name, 112, *112*, **223**, *223*
—.ordinal, 191, **336**, *336*
—._ordinal, 112, **223**
CITATION_EDITOR, 83, 112, 190, 191, **224**, **337**
—.citation_id, 191, **337**, *337*
—._citation_id, 112, **224**, *224*
—.name, 191, **337**, *337*
—._name, 112, **224**, *224*
—.ordinal, 191, **337**
—._ordinal, 112, **224**
COMPUTING, 91, 113, 190, 191, **224**, **337**, 416
—.cell_refinement, 191, **337**, *337*
—._cell_refinement, *6*, 113, **224**, *224*, 565
—.data_collection, 191, **337**, *337*
—._data_collection, *6*, 113, **224**, *224*, 565
—.data_reduction, 191, **337**, *337*
—._data_reduction, *6*, 113, **224**, *224*, 565
—.entry_id, 191, **337**
—.molecular_graphics, 191, **337**, *337*
—._molecular_graphics, *6*, 113, **224**, *224*, 565
—.publication_material, 191, **337**, *337*
—._publication_material, *6*, 113, **224**, *224*, 565
—.structure_refinement, 191, **337**, *337*
—._structure_refinement, *6*, 113, **224**, *224*, 565
—.structure_solution, 191, **337**, *337*
—._structure_solution, *6*, 98, 113, **224**, *224*, 565

DATABASE, 113, 190–192, 195, **224**, **337**, 338
—.code_CAS, 192, **337**
—._code_CAS, 84, 113, **224**
—.code_CSD, 192, **337**
—._code_CSD, 84, *84*, 113, **224**, *224*
—.code_depnum_ccdc_archive, 192, **338**
—._code_depnum_ccdc_archive, 113, **224**
—.code_depnum_ccdc_fiz, 192, **338**
—._code_depnum_ccdc_fiz, 113, **224**
—.code_depnum_ccdc_journal, 192, **338**
—._code_depnum_ccdc_journal, 113, **224**
—.code_ICSD, 192, **337**
—._code_ICSD, 113, **224**
—.code_MDF, 192, **338**
—._code_MDF, 113, **224**
—.code_NBS, 192, **338**
—._code_NBS, 113, **224**
—.code_PDB, 192, **338**
—._code_PDB, 84, 113, **224**
—.code_PDF, 192, **338**
—._code_PDF, 113, **224**
—.CSD_history, 192, **338**
—._CSD_history, 113, **224**
—.entry_id, 192, **338**
—.journal_ASTM, 192, **338**
—._journal_ASTM, 113, **224**
—.journal_CSD, 192, **338**
—._journal_CSD, 113, **224**
DATABASE_2, 84, 113, 186, 190–192, 195, **338**
—.database_code, 84, *84*, 192, **338**, *338*
—.database_id, 84, *84*, 192, **338**, *338*
DATABASE_PDB_CAVEAT, 192, 193, 195, **338**
—.id, 192, 193, **339**, *339*
—.text, 192, *193*, **339**, *339*
DATABASE_PDB_MATRIX, 192, 193, 195, **339**
—.entry_id, 192, **339**
—.origx[m][n] ($m, n = 1, 2, 3$), 192, **339**
—.origx_vector[n] ($n = 1, 2, 3$), 192, **339**
—.scale[m][n] ($m, n = 1, 2, 3$), 192, **339**
—.scale_vector[n] ($n = 1, 2, 3$), 192, **339, 340**
DATABASE_PDB_REMARK, 192, 193, 195, **340**
—.id, 192, 193, **340**, *340*
—.text, 192, *193*, **340**, *340*
DATABASE_PDB_REV, 192, 193, 195, **340**, 341
—.author_name, 192, **340**, *340*
—.date, 192, **340**, *340*
—.date_original, 192, **340**, *340*
—.mod_type, 192, **340**, *340*
—.num, 192, **340**, *340*, 341
—.replaced_by, 192, **340**
—.replaces, 192, **341**
—.status, 192, *340*, **341**
DATABASE_PDB_REV_RECORD, 192, 193, 195, **341**
—.details, 192, **341**, *341*
—.rev_num, 192, **341**, *341*
—.type, 192, **341**, *341*
DATABASE_PDB_TVECT, 192, 193, 195, **341**
—.details, 192, **341**
—.id, 192, **341**
—.vector[1], 192, **341**
—.vector[2], 192, **341**
—.vector[3], 192, **341**
_datablock.description, *see* DDL2
_datablock.id, *see* DDL2
DATABLOCK, 70, **474**
DATABLOCK_METHODS, 70, **474**
DDL0 (informal Dictionary Definition Language)
_compliance, *7*
_definition, *7*
_enumeration, *7*
_enumeration_default, *7*
_enumeration_detail, *7*
_enumeration_range, *7*
_esd, *7*
_esd_default, *7*
_example, *7*
_example_detail, *7*
_list, *7*
_list_identifier, *7*
_name, *7*
_type, *7*
_units_conversion, *7*
_units_description, *7*
_units_extension, *7*
_update_history, *7*
DDL1 (Dictionary Definition Language)
_category, 47, 55, *55–58*, 59, 75, *75*, 76, *76*, 77, *77*, 78, *78*, *89*, **471**, 472
_definition, 55, *55–57*, 58, *58, 59*, *75–77*, 78, *78*, *89*, **471**
_dictionary_history, *55*, 60, *76, 77*, 89, **471**
_dictionary_name, *55*, 60, 76, *76, 77*, 85, 87–89, **471**
_dictionary_update, *55*, 60, *76, 77*, 89, **471**
_dictionary_version, *55*, 60, *76, 77*, 85, 88, 89, **471**
_enumeration, 51, 57, 58, *58*, *78*, 105, 471, **471**, 472
_enumeration_default, 56, *56*, 57, 58, *89*, **471**
_enumeration_detail, 57, 58, *58*, **471**
_enumeration_range, 51, 55, *55, 56*, 58, *75*, *78*, *89*, 212, 243, **471**
_example, *57*, 58, *89*, **471**, 490, *490, 491*
_example_detail, *57*, 58, *76, 77*, *89*, **471**
_list, 46, *55*, 56, *56, 57*, 58, *58, 59*, *75*, *78*, 89, *89*, **471**
DDL1 (Dictionary Definition Language)
_list_level, 58, *59*, **471**
_list_link_child, 53, 56, *57*, 59, 78, 88, **471**
_list_link_parent, 53, 56, *57*, 59, *75*, **471**
_list_mandatory, 56, *57*, 59, 60, 75, *75*, *78*, 81, **471**, 472
_list_reference, 54, *55*, 56, *56–58*, 59, 60, 75, *75*, 76, 78, *78*, 81, *89*, **471**
_list_uniqueness, 59, 60, 75, 81, **472**
_name, 55, *55*, 56, *56, 57*, 58, *58, 59*, *75–78*, 88, 89, *89*, 259, **472**, 520
_related_function, 56, *57*, 59, 60, 471, **472**
_related_item, 56, *57*, 59, 60, 471, **472**
_type, 55, *55*, 56, *56, 57*, 58, *58, 59*, *75, 76*, 77, *77, 78*, *89*, **472**, 520
_type_conditions, *34*, 50, *55*, 56, *56, 57*, 58, 59, *75*, *78*, 148, **472**
_type_construct, 58, 59, *59*, 85, **472**
_units, *55*, 56, *56, 57*, 58, 59, *75*, 78, *78*, 79, **472**
_units_detail, *55*, 56, *56, 57*, 58, 59, *75*, *78*, 79, **472**
DDL2 (Dictionary Definition Language)
_category.description, *65, 66*, 68, *81*, **473**
_category.id, *65, 66*, 68, *81*, **473**
_category.implicit_key, 68, 70, **473**
_category.mandatory_code, 64, *65, 66*, 68, *81*, **473**
_category_examples.case, 64, *65*, 69, *81*, **473**
_category_examples.detail, 64, *65*, 69, *81*, **473**
_category_examples.id, 69
_category_group.category_id, 69
_category_group.id, *65, 66*, 80, *81*
_category_group_list.description, 69, *69*, *80*, 81, **473**
_category_group_list.id, *69*, *80*, 81, **473**
_category_group_list.parent_id, 69, *69*, *80*, 81, **473**
_category_key.id, 69
_category_key.name, 54, *65, 66*, 69, *81*, **473**
_datablock.description, 70, **474**
_datablock.id, 70, **474**
_dictionary.datablock_id, 70, 79, *80*, **474**
_dictionary.title, 70, 79, *80*, 87, **474**
_dictionary.version, 70, *80*, **474**
_dictionary_history.revision, *80*, **474**
_dictionary_history.update, *80*, **474**
_dictionary_history.version, *80*, **474**
_item.category_id, *55*, *61*, *64*, 65, *66*, 80, *82, 83*
_item.mandatory_code, *55*, *61*, *64*, 66, *66*, 82, *82, 83*, **474**
_item.name, *55*, *61*, 63, *64*, 65, 66, *66*, 67, 82, *82, 83*, 88, **474**, 520
_item_aliases.alias_name, *55*, *61*, *64*, *66*, *83*, 144, **474**, 520
_item_aliases.dictionary, *55*, *61*, *64*, 66, *66*, *83*, **474**
_item_aliases.name, 66
_item_aliases.version, *55*, *61*, *64*, 66, *66*, *83*, **474**
_item_default.name, 66
_item_default.value, *64*, 66, **475**
_item_dependent.dependent_name, *55*, *64*, 66, 82, **475**
_item_dependent.name, 54
_item_description.description, *55*, *61*, *64*, 66, *66*, 82, 83, *83*, **475**
_item_description.name, 63, 66
_item_enumeration.detail, 67, **475**
_item_enumeration.name, 67
_item_enumeration.value, 67, **475**
_item_examples.case, *64*, *66*, 67, **475**
_item_examples.detail, 67, **475**
_item_linked.child_name, 66, *82*, **475**
_item_linked.parent_name, 66, *66*, 67, *82*, **475**, 520
_item_range.maximum, *64*, 67, *83*, **475**
_item_range.minimum, *64*, 67, *83*, **475**
_item_related.function_code, *55*, 64, *64*, 65, 67, *83*, **476**
_item_related.name, 476

DDL2 (Dictionary Definition Language)
_item_related.related_name, *55*, *64*, 65, 67, *83*, 476
_item_structure.code, 67
_item_structure.organization, 67, **476**
_item_structure_list.code, 67, **476**
_item_structure_list.dimension, 67, **476**
_item_structure_list.index, 67, **476**
_item_sub_category.id, *55*, *64*, 82
_item_type.code, *55*, *61*, *64*, *66*, 67, 69, *82, 83*, 476, 520
_item_type.name, 68, 69
_item_type_conditions.code, 23, *55*, *64*, *83*, **476**
_item_type_list.code, 67, 68, *68*, *80*, 85, **476**
_item_type_list.construct, 68, *68*, *80*, **477**
_item_type_list.detail, *68*, *80*, **477**
_item_type_list.primitive_code, 68, *68*, *80*, **477**
_item_units.code, *55*, 64, *64*, 68, 83, *83*
_item_units.name, 68
_item_units_conversion.factor, 68, *80*, **477**
_item_units_conversion.from_code, 68, *80*, **477**
_item_units_conversion.operator, 68, *80*, **477**
_item_units_conversion.to_code, 68, *80*, **477**
_item_units_list.code, 68, *80*, 83, **477**
_item_units_list.detail, *80*, 83, **477**
_method_list.code, 70, **477**
_method_list.detail, 70, **477**
_method_list.id, 70, **477**
_method_list.inline, 70, **477**
_method_list.language, 70, **477**
_sub_category.description, 80, *80*, **478**
_sub_category.id, 80, *80*, **478**
_sub_category_examples.case, 70, **478**
_sub_category_examples.detail, 70, **478**
_sub_category_examples.id, 70
_sub_category_methods.method_id, 70
_sub_category_methods.sub_category_id, 70
_define_bonding_convention, *see* MIF
_define_stereo_relationship, *see* MIF
_definition, *see* DDL1
_dictionary.datablock_id, *see* DDL2
_dictionary_history, *see* DDL1
_dictionary_history.revision, *see* DDL2
_dictionary_history.update, *see* DDL2
_dictionary_history.version, *see* DDL2
_dictionary_name, *see* DDL1
_dictionary.title, *see* DDL2
_dictionary_update, *see* DDL1
_dictionary.version, *see* DDL2
_dictionary_version, *see* DDL1
DICTIONARY, 70, **474**
DICTIONARY_HISTORY, 70, **474**
DIFFRN, 77, 92, 93, 133, 148–151, 157, 199, 201–203, **225**, **341**, 343–345, 347, 348, 351, 352, 450–452, 552
—.ambient_environment, 149, *151*, **341**, *341*
—._ambient_environment, 93, 94, **225**
—.ambient_pressure, 149, **341**, 342
—._ambient_pressure, 93, 118, **225**, 266
—.ambient_pressure_esd, **342**
—.ambient_pressure_gt, 149, **342**
—._ambient_pressure_gt, 93, **225**
—.ambient_pressure_lt, 149, **342**
—._ambient_pressure_lt, 93, **225**
—.ambient_temp, 149, *151*, *341*, **342**
—.ambient_temp_details, 149, **342**
—.ambient_temp_esd, **342**
—.ambient_temp_gt, 149, **342**
—.ambient_temp_lt, 149, **342**
—._ambient_temperature, 93, 94, 118, **225**, *225*, 266
—._ambient_temperature_gt, 93, **225**
—._ambient_temperature_lt, 93, 94, **225**
—.crystal_id, *38*, 149, *151*, **342**, *454, 455*, *553*
—.crystal_support, 149, *151*, *341*, **342**
—.crystal_treatment, 149, *151*, *341*, **342**
—._crystal_treatment, 93, 94, **225**
—.details, 149, *341*, **342**
DIFFRN
—.id, *38*, 149, 150, *151*, 203, *341*, **342**, 343–345, 347, 348, 351, 352, 450–452, *454, 455*, 550, *553*
—._measured_fraction_theta_full, 93, **225**, 565, 567
—._measured_fraction_theta_max, 93, **225**, 565, 567
—._special_details, 93, 94, **225**, *225*
—._symmetry_description, 93, *137, 138*, **225**
DIFFRN_ATTENUATOR, 93, 94, 149, **225**, **342**
—.code, 149, **342**, *342*, 343, 347
—._code, 94, *94*, **225**, *225*, 228
—.material, 149, **342**
—._material, 94, *94*, **225**
—.scale, 149, 342, *342*, **343**
—._scale, 94, *94*, **225**, *225*
DIFFRN_DATA_FRAME, 202, 203, **449**, 451, 457, 458
—.array_id, 203, **449**, *449*, *454*, *456*, 550, *553*
—.binary_id, 203, **449**, *449*, *454*, *456*, *553*
—.detector_element_id, 203, **449**, *449*, *454*, *456*, *553*
—.id, 203–205, **449**, *449*, 453, *454*, *456*, 457, 458, *553*
DIFFRN_DETECTOR, 93–95, 149, 202, 203, **225**, 227, **343**, 345, 448, **449**, 450, 452, 550, 552
_diffrn_radiation_detector, 95, **226**
_diffrn_radiation_detector_dtime, 95, **226**
—, 95, **225**, *225*
—.area_resol_mean, 149, **343**
—._area_resol_mean, 95, **226**, 565
—.details, 149, 203, **343**, **449**, *553*
—._details, 95, **226**
—.detector, *38*, 149, *151*, 203, **343**, *343*, *449*, **450**
—.diffrn_id, *38*, 149, 150, *151*, 203, **343**, *343*, *449*, **450**, *454, 455*, *553*
—.dtime, 149, 203, **343**, **450**
—._dtime, 95, 119, 126, **226**
—.id, *38*, 203, **450**, *454, 455*, *553*
—.number_of_axes, 203, **450**, *454, 455*, *553*
—.type, *38*, 149, *151*, 203, **343**, *343*, *449*, **450**, *454, 455*, *553*
—._type, 95, 120, 126, *225*, **226**
DIFFRN_DETECTOR_AXIS, 199, 202–204, 447, **450**, 552
—.axis_id, 203, **450**, *454, 455*, *553*
—.detector_id, 203, **450**, *454, 455*, *553*
—.id, **450**
DIFFRN_DETECTOR_ELEMENT, 202, 203, 449, **450**, 451, 550, 552
—.center[1], 203, **450**, *450*
—.center[2], 203, **450**, *450*
—.detector_id, *38*, 203, **450**, *450*, *454, 455*, *553*
—.id, *38*, 203, 449, **450**, *450*, 451, *454, 455*, 550, *553*
DIFFRN_FRAME_DATA, 449, **451**, 555
—.array_id, *38*, **451**
—.binary_id, *38*, **451**
—.detector_element_id, *38*, **451**
—.id, *38*, **451**
DIFFRN_MEASUREMENT, 93, 95, 149, 202, 203, **226**, **343**, **451**, 452, 550, 551
—.details, 149, *151*, 203, **343**, *343*, **451**, *451*, *553*
—._details, 95, **226**
—.device, 149, *151*, 203, **343**, *343*, **451**, *451*, 452
—._device, *6*, 95, **226**
—.device_details, 149, *151*, 203, **343**, *343*, **451**, *451*
—._device_details, 95, **226**
—.device_type, 149, *151*, 203, **343**, *343*, **451**, *451*
—._device_type, 95, **226**, *226*, 565
—.diffrn_id, *38*, 149, *151*, 203, **343**, *343*, **451**, *451*, *454*, *456*, *553*
—.id, 203, **451**, 452, *454*, *456*, *553*
—.method, *38*, 149, *151*, 203, **343**, *343*, **451**, *451*, *454*, *456*, *553*
—._method, *6*, 95, *95*, **226**, *226*, 565
DIFFRN_MEASUREMENT
—.number_of_axes, 203, **451**, *454*, *456*, *553*
—.specimen_support, 149, 203, **344**, **452**
—._specimen_support, 95, **226**
DIFFRN_MEASUREMENT_AXIS, 199, 202–204, 451, **452**, 551
—.axis_id, 203, **452**, *454*, *456*, *553*
—.id, **452**
—.measurement_device, 203, **452**
—.measurement_id, 203, **452**, *454*, *456*, *553*
DIFFRN_ORIENT_MATRIX, 93, 95, 149, **226**, **344**
—.diffrn_id, 149, **344**, *344*
—.type, 149, **344**, *344*
—._type, 95, **226**, *226*
—.UB[m][n] ($m, n = 1, 2, 3$), 149, **344**, *344*
—._UB_mn ($m, n = 1, 2, 3$), 95, **226**, *226*
DIFFRN_ORIENT_REFLN, 93, 95, 149, **227**, **344**, 345
—.angle_chi, 149, *344*, **345**
—._angle_chi, 95, **227**
—.angle_kappa, 149, *344*, **345**
—._angle_kappa, 95, **227**, *227*
—.angle_omega, 149, *344*, **345**
—._angle_omega, 95, **227**, *227*
—.angle_phi, 149, *344*, **345**
—._angle_phi, 95, **227**, *227*
—.angle_psi, 149, *344*, **345**
—._angle_psi, 95, **227**
—.angle_theta, 149, *344*, **345**
—._angle_theta, 95, **227**, *227*
—.diffrn_id, 149, *344*, **345**
—.index_h, 149, *344*, **345**
—._index_h, 95, **227**, *227*
—.index_k, 149, *344*, **345**
—._index_k, 95, **227**, *227*
—.index_l, 149, *344*, **345**
—._index_l, 95, **227**, *227*
DIFFRN_RADIATION, 93–95, 101, 149, 202, 204, **227**, 228, **345**, 348, 409, **452**, 549, 550
—.collimation, 149, *151*, 204, **345**, *345*, **452**, *452*, *553*
—._collimation, 94, **227**
—.diffrn_id, *38*, 149, *151*, 204, **345**, *345*, **452**, *452*, *454, 455*, *553*
—.div_x_source, 204, **452**, *454, 455*, 550, *553*
—.div_x_y_source, 204, **452**, *454, 455*, 550, *553*
—.div_y_source, 204, **452**, *454, 455*, 550, *553*
—.filter_edge, 149, 204, **345**, **452**
—._filter_edge, 94, **227**
—.inhomogeneity, 149, 204, **345**, **452**
—._inhomogeneity, 94, **227**
—.monochromator, 149, *151*, 204, **345**, *345*, **452**, *452*, *454, 455*, *553*
—._monochromator, 94, **227**, *227*
—.polarisn_norm, 149, 204, **345**, **452**, 453
—._polarisn_norm, 94, **227**
—.polarisn_ratio, 149, 204, **345**, **453**
—._polarisn_ratio, 94, **227**, 261, 452
—.polarizn_source_norm, 204, **453**, *454, 455*, 549, *553*
—.polarizn_source_ratio, 204, **453**, *454, 455*, 549, *553*
—.probe, 149, 204, **346**, **453**, *553*
—._probe, 94, 120, *125*, 129, **227**, 261
—.type, 149, *151*, 204, *345*, **346**, *452*, **453**
—._type, *6*, 94, 120, **227**, *227*, 261, *563*, 565, 566
—.wavelength_id, *38*, 150, *151*, 204, *345*, **346**, 348, 409, *452*, **453**, *454, 455*, 549, *553*
—.xray_symbol, 150, 204, **346**, **453**
—._xray_symbol, 94, **227**
DIFFRN_RADIATION_WAVELENGTH, 93–95, 149, 150, **227**, 319, **346**, 453, 549
—, *6*, 94, 129, 215, 217, *227*, **228**, 261, 565, 566
—.id, *38*, 150, *151*, 313, **346**, *346*, 453, *454, 455*, *553*
—._id, 94, 95, 122, **228**, 250, 261, 265, 268
—.wavelength, *38*, 150, *151*, **346**, *346*, *454, 455*, 549, *553*

DIFFRN_RADIATION_WAVELENGTH
—.wt, *38*, 150, *151*, **346**, *346*, *454, 455*, *553*
—_wt, 94, **228**, 261
DIFFRN_REFLN, 93, 95–97, 100, 101, 132, 133, 149–151, 202, 204, **228**, 229, **285**, **346**, 347–351, 411, 413, **453**
—.angle_chi, 150, 204, **346**, *346*
—_angle_chi, *94*, 96, *96*, **228**, *228*
—.angle_kappa, 150, 204, **346**, *346*
—_angle_kappa, 96, *96*, **228**, *228*
—.angle_omega, 150, 204, *346*, **347**
—_angle_omega, 96, *96*, **228**, *228*
—.angle_phi, 150, 204, *346*, **347**
—_angle_phi, 96, *96*, **228**, *228*
—.angle_psi, 150, 204, *346*, **347**
—_angle_psi, 96, **228**
—.angle_theta, 150, 204, *346*, **347**
—_angle_theta, 96, *96*, **228**, *228*
—.attenuator_code, 150, 204, 342, *346*, **347**
—_attenuator_code, 94, *94*, 96, 225, **228**
—.class_code, 150, **347**
—_class_code, 96, 97, **228**
—.counts_bg_1, 150, 204, *346*, **347**
—_counts_bg_1, *94*, 96, *96*, **228**, *228*
—.counts_bg_2, 150, 204, *346*, **347**
—_counts_bg_2, *94*, 96, *96*, **228**, *228*
—.counts_net, 150, 204, *346*, **347**
—_counts_net, 96, **228**
—.counts_peak, 150, 204, *346*, **347**
—_counts_peak, 96, **228**
—.counts_total, 150, 204, *346*, **347**
—_counts_total, *94*, 96, *96*, **228**, *228*
—_crystal_id, 96, **228**, 235
—.detect_slit_horiz, 150, 204, *346*, **347**
—_detect_slit_horiz, 96, **228**
—.detect_slit_vert, 150, 204, *346*, **347**
—_detect_slit_vert, 96, **228**
—.diffrn_id, 150, 151, 204, 346, *346*, **347**
—.elapsed_time, 150, 204, *346*, **347**
—_elapsed_time, *94*, 96, *96*, 97, **228**, *228*
—.frame_id, 204, **453**
—.id, 150, 151, 204, *346*, **347**
—.index_h, 82, 150, 204, *346*, **347**, 348, 349
—_index_h, 82, 84, *94*, 95, *96*, 151, 217, *228*, **229**, 230
—.index_k, 82, 150, 204, *346*, **347**, 349
—_index_k, 82, 84, *94*, 95, *96*, *228*, **229**
—.index_l, 82, 150, 204, *346*, **347**, 349
—_index_l, 82, 84, *94*, 95, *96*, *228*, **229**
—_index_m_*n* ($n = 1, \ldots, 8$), 133, **285**
—.intensity_net, 150, 204, *346*, **348**
—_intensity_net, 96, **229**
—.intensity_sigma, 150, 204, *346*, **348**
—_intensity_sigma, 96, **229**
—.intensity_u, 150, **348**
—_intensity_u, 96, **229**
—.scale_group_code, 150, 204, *346*, **348**
—_scale_group_code, 96–98, **229**
—.scan_mode, 150, 204, *346*, **348**
—_scan_mode, 96, **229**
—.scan_mode_backgd, 150, 204, *346*, 348, **348**
—_scan_mode_backgd, 96, **229**
—.scan_rate, 150, 204, *346*, **348**
—_scan_rate, *94*, 96, *96*, *228*, **229**
—.scan_time_backgd, 150, 204, *346*, **348**
—_scan_time_backgd, 96, **229**
—.scan_width, 150, 204, *346*, **348**
—_scan_width, *94*, 96, *96*, *228*, **229**
—_sint/lambda, 96, **229**
—.sint_over_lambda, 150, 204, *346*, **348**
—.standard_code, 150, 204, *346*, **348**
—_standard_code, 96, 97, **229**, 232
—.wavelength, 150, 204, *346*, **348**
—_wavelength, 96, **229**
—.wavelength_id, 150, 204, 346, *346*, **348**
—_wavelength_id, 95, 96, **229**
DIFFRN_REFLNS, 93, 96, 97, 100, 132, 133, 149, 150, 202, 228, **229**, **285**, 346, **348**
—.av_R_equivalents, 150, **348**
DIFFRN_REFLNS
—_av_R_equivalents, *6*, 96, *229*, **230**, 565, 566
—_av_sigmaI/netI, 96, **230**
—.av_sigmaI_over_netI, 150, **348**
—.av_unetI/netI, 150, **348**
—_av_unetI/netI, 96, *229*, **230**
—.diffrn_id, 150, 346, **348**
—.limit_h_max, 150, **348**, 411
—_limit_h_max, *6*, 96, *137*, *229*, **230**, 565
—.limit_h_min, 150, **349**, 411
—_limit_h_min, *6*, 96, *137*, *229*, **230**, 565
—_limit_index_m_*n*_max ($n = 1, \ldots, 8$), 133, *137*, **285**
—_limit_index_m_*n*_min ($n = 1, \ldots, 8$), 133, *137*, **285**
—.limit_k_max, 150, **349**, 411
—_limit_k_max, *6*, 96, *137*, *229*, **230**, 565
—.limit_k_min, 150, **349**, 411
—_limit_k_min, *6*, 96, *137*, *229*, **230**, 565
—.limit_l_max, 150, **349**, 411
—_limit_l_max, *6*, 96, *137*, *229*, **230**, 565
—.limit_l_min, 150, **349**, 411
—_limit_l_min, *6*, 96, *137*, *229*, **230**, 565
—.number, 150, **349**
—_number, *6*, 96, 97, *229*, **230**, 565–567
—.reduction_process, 150, 347, **349**
—_reduction_process, 96, 229, **230**
—_resolution_full, 96, **230**
—_resolution_max, 96, **230**
—_satellite_order_max, 133, *137*, **285**
—_theta_full, 94, 96, 225, **230**, 565, 567
—.theta_max, 150, **349**
—_theta_max, *6*, 96, *137*, 225, *229*, **230**, 565, 567
—.theta_min, 150, **349**
—_theta_min, 96, *137*, *229*, **230**
—.transf_matrix[*m*][*n*] ($m, n = 1, 2, 3$), 150, **349**
—_transf_matrix_11, 96, **230**
—_transf_matrix_12, 96, **230**
—_transf_matrix_13, 96, **230**
—_transf_matrix_21, 96, **230**
—_transf_matrix_22, 96, **230**
—_transf_matrix_23, 96, **230**
—_transf_matrix_31, 96, **230**
—_transf_matrix_32, 96, **230**
—_transf_matrix_33, 96, **230**
DIFFRN_REFLNS_CLASS, 93, 96, 97, 149, 150, 164, **230**, **350**
—.av_R_eq, 150, **350**, *350*
—_av_R_eq, 96, *97*, **230**, *230*
—.av_sgI/I, 150, **350**
—_av_sgI/I, 96, **230**
—.av_uI/I, 150, **350**
—_av_uI/I, 96, **231**
—.code, 150, **350**, *350*
—_code, 96, *97*, *230*, **231**
—.d_res_high, 150, **350**, *350*
—_d_res_high, 96, *97*, *230*, **231**
—.d_res_low, 150, **350**, *350*
—_d_res_low, 96, *97*, *230*, **231**
—.description, 150, **350**, *350*
—_description, 96, *97*, *230*, **231**
—.number, 150, **350**, *350*
—_number, 96, *97*, *230*, **231**
DIFFRN_SCALE_GROUP, 93, 96–98, 150, **231**, 348, **350**
—.code, 150, 348, **350**, *350*
—_code, 96, 97, *97*, 229, **231**, *231*
—.I_net, 150, **350**, *350*
—_I_net, 96, 97, *97*, **231**, *231*
DIFFRN_SCAN, 202, 204, 205, **453**, 458
—.date_end, 204, *453*, **457**
—.date_start, 204, 205, *453*, **457**
—.frame_id_end, 204, *453, 454*, *456*, **457**, *553*
—.frame_id_start, 204, 205, *453, 454*, *456*, **457**, *553*
—.frames, 204, 205, *453, 454*, *456*, **457**, *553*
—.id, 204, 205, *453, 454*, *456*, **457**, 458, *553*
—.integration_time, 204, 205, *453*, **457**
DIFFRN_SCAN_AXIS, 199, 202, 204, 205, **457**, 458, 551
—.angle_increment, 204, *453, 454*, *456*, **457**, *553*
—.angle_range, 204, *453, 454*, *456*, **457**, *553*
—.angle_rstrt_incr, 204, **457**
—.angle_start, 204, *453, 454*, *456*, **457**, *553*
—.axis_id, 204, 205, *453, 454*, *456*, **457**, *553*
—.displacement_increment, 204, *453, 454*, *456*, **457**, *553*
—.displacement_range, 204, *453, 454*, *456*, **457**, *553*
—.displacement_rstrt_incr, 204, **457**
—.displacement_start, 204, *453, 454*, *456*, **458**, *553*
—.scan_id, 204, 205, *453, 454*, *456*, **458**, *553*
DIFFRN_SCAN_FRAME, 202, 204, 205, **458**, 551
—.date, 204, 205, *454*, *456*, **458**, *553*
—.frame_id, 204, 205, *454*, *456*, **458**, *553*
—.frame_number, 204, 205, *454*, *456*, **458**, *553*
—.integration_time, 204, 205, *454*, *456*, 457, **458**, 551, *553*
—.scan_id, 204, *454*, *456*, **458**, *553*
DIFFRN_SCAN_FRAME_AXIS, 202, 204, 205, **458**, 551
—.angle, 204, 205, *454*, *456*, 457, **458**, *553*
—.angle_increment, 205, *454*, 457, **458**
—.angle_rstrt_incr, 205, 457, **458**
—.axis_id, 204, 205, *454*, *456*, **458**, *553*
—.displacement, 205, *454*, *456*, 457, **458**, *553*
—.displacement_increment, 205, *454*, **458**
—.displacement_rstrt_incr, 205, **458**
—.frame_id, 204, 205, *454*, *456*, **458**, *553*
DIFFRN_SOURCE, 93–95, 149, 150, 202, **231**, **350**, 550
_diffrn_radiation_source, 94, 95, **231**
—, 94, 95, **231**, *231*, 261
—.current, 150, *151*, **350**, *350*, *553*
—_current, 94, **231**, *231*
—.details, 150, **351**
—_details, 94, 95, **231**
—.diffrn_id, *38*, 150, *151*, *350*, **351**, *454, 455*, *553*
—.power, 150, *151*, *350*, **351**
—_power, 94, **231**, *231*
—.size, 150, *350*, **351**
—_size, 94, **231**, *231*
—.source, *38*, 150, *151*, *350*, **351**, *454, 455*, *553*
—.take-off_angle, 150, **351**
—_take-off_angle, 94, **232**
—.target, 150, *151*, **351**
—_target, 94, 95, **232**
—.type, *38*, 150, *151*, *350*, **351**, *454, 455*, *553*
—_type, 94, *231*, **232**
—.voltage, 150, **351**
—_voltage, 94, **232**
DIFFRN_STANDARD_REFLN, 93, 96, 97, 132, 133, 149, 150, **232**, **285**, 348, **351**
—.code, 150, 348, **351**, *351*
—_code, 96, 97, 229, **232**
—.diffrn_id, 150, **351**, *351*
—.index_h, 150, **351**, *351*
—_index_h, 96, **232**, *232*
—.index_k, 150, **351**, *351*
—_index_k, 96, **232**, *232*
—.index_l, 150, **351**, *351*
—_index_l, 96, **232**, *232*
—_index_m_*n* ($n = 1, \ldots, 8$), 133, **285**
DIFFRN_STANDARDS, 93, 96, 97, 149, 150, **232**, **352**
—.decay_%, 150, **352**
—_decay_%, *6*, 96, **232**, 565
—.diffrn_id, 150, **352**, *352*
—.interval_count, 150, **352**
—_interval_count, *6*, 96, **232**, 565
—.interval_time, 150, **352**, *352*
—_interval_time, *6*, 96, **232**, *232*, 565
—.number, 150, **352**, *352*
—_number, *6*, 96, **232**, *232*, 565

DIFFRN_STANDARDS
—.scale_sigma, 150, **352**
—._scale_sigma, 96, **232**
—.scale_u, 150, **352**
—._scale_u, 96, **232**
_display_colour, *see* MIF
_display_conn_colour, *see* MIF
_display_conn_id, *see* MIF
_display_conn_symbol, *see* MIF
_display_coord_x, *see* MIF
_display_coord_y, *see* MIF
_display_define_origin, *see* MIF
_display_define_scale, *see* MIF
_display_define_span_x, *see* MIF
_display_define_span_y, *see* MIF
_display_id, *see* MIF
_display_object, *see* MIF
_display_size, *see* MIF
_display_symbol, *see* MIF

ENTITY, 91, 146, 164, 166, 174–176, 180, 302, 330, 332, 333, **352**, 353–356, 418, 433
—.details, *5*, 174, *175*, **352**, *352*
—.formula_weight, *145*, 174, *175*, **352**, *352*
—.id, *5*, *145*, 166, 174, *175, 176*, 302, **352**, *352*, 353–356, 418, 433
—.src_method, *145*, 174, *176*, **353**
—.type, *5*, *145*, 174, *175, 176*, *352*, **353**
ENTITY_KEYWORDS, 174, 176, **353**
—.entity_id, 174, *176*, **353**, *353*
—.text, 174, *176*, **353**, *353*
ENTITY_LINK, 169, 173, 174, 182, **353**
—.details, 173, **353**
—.entity_id_1, 173, **353**
—.entity_id_2, 173, **353**
—.entity_seq_num_1, 173, **353**
—.entity_seq_num_2, 173, **353**
—.link_id, 173, 174, **353**
ENTITY_NAME_COM, 174–176, **353**
—.entity_id, *146*, 174, *175, 176*, **353**, *353*
—.name, *146*, 174, *175, 176*, **353**, *353*
ENTITY_NAME_SYS, 174, 175, **354**
—.entity_id, 174, *175*, **354**, *354*
—.name, 174, *175*, **354**, *354*
—.system, 174, 175, *175*, **354**
ENTITY_POLY, 146, 174–176, 353, **354**
—.entity_id, *146*, 175, *176*, **354**, *354*
—.nstd_chirality, *146*, 175, 176, **354**, *354*
—.nstd_linkage, *146*, 175, 176, **354**, *354*
—.nstd_monomer, *146*, 175, 176, **354**, *354*
—.number_of_monomers, 175, *176*, **354**
—.type, *146*, 175, *176*, **354**, *354*
—.type_details, *146*, 175, **354**, *354*
ENTITY_POLY_SEQ, 166, 174–176, 302, 353, **354**, 355, 434
—.entity_id, *5*, *146*, 175, *176*, *354*, **355**
—.hetero, 175, **355**
—.mon_id, *5*, 146, *146*, 175, 176, *176*, *354*, **355**
—.num, *5*, *146*, 166, 175, *176*, 302, 353, *354*, **355**, 434
ENTITY_SRC_GEN, 174, 175, 195, 353, **355**
—.entity_id, 174, *175*, **355**, *355*
—.gene_src_common_name, 174, *175*, **355**, *355*
—.gene_src_details, 174, **355**
—.gene_src_genus, 174, **355**, *355*
—.gene_src_species, 174, **355**, *355*
—.gene_src_strain, 174, *175*, **355**, *355*
—.gene_src_tissue, 174, **355**
—.gene_src_tissue_fraction, 174, **355**
—.host_org_common_name, 174, *175*, **355**, *355*
—.host_org_details, 174, **356**
—.host_org_genus, 174, *175*, *355*, **356**
—.host_org_species, 174, *175*, *355*, **356**
—.host_org_strain, 174, **356**
—.plasmid_details, 174, **356**
—.plasmid_name, 174, *175*, *355*, **356**
ENTITY_SRC_NAT, 174, 175, 195, 353, **356**
—.common_name, *146*, 174, **356**, *356*
—.details, *146*, 174, **356**, *356*
ENTITY_SRC_NAT
—.entity_id, *146*, 174, **356**, *356*
—.genus, *146*, 174, **356**, *356*
—.species, *146*, 174, **356**, *356*
—.strain, 174, **356**
—.tissue, 174, **356**
—.tissue_fraction, 174, **356**
ENTRY, 194, 195, 308, 316, 318, 331, 334, 337–339, **356**, 357, 362, 373, 374, 377, 387, 389, 391, 398, 411, 418, 424, 442
—.id, *5*, *38*, 64, 194, 195, 308, 316, 318, 331, 334, 337–339, **356**, *356*, 357, 362, 373, 374, 377, 387, 389, 391, 398, 411, 418, 424, 442
ENTRY_LINK, 194, 195, **357**
—.details, 194, **357**, *357*
—.entry_id, 194, **357**, *357*
—.id, 194, **357**, *357*
_enumeration, *see* DDL1
_enumeration_default, *see* DDL1
_enumeration_detail, *see* DDL1
_enumeration_range, *see* DDL1
_example, *see* DDL1
_example_detail, *see* DDL1
EXPTL, 76, 77, 90, 97, 98, 133, 134, 148, 151, 152, **233**, 356, **357**
—.absorpt_coefficient_mu, *81*, 151, **357**, *357*
—._absorpt_coefficient_mu, *6*, *77*, 97, 118, **233**, *233*, 260, *563*, 565
—.absorpt_correction_T_max, *81*, 151, **357**, *357*
—._absorpt_correction_T_max, *6*, *77*, 97, **233**, *233*, 565
—.absorpt_correction_T_min, *81*, 151, **357**, *357*
—._absorpt_correction_T_min, *6*, *77*, 97, **233**, *233*, 565
—.absorpt_correction_type, *81*, 151, **357**, *357*
—._absorpt_correction_type, *6*, *77*, 97, **233**, *233*, 565, 566
—.absorpt_process_details, 81, 151, **357**, *357*
—._absorpt_process_details, *77*, 97, **233**, *233*, 565
—.crystals_number, *81*, 151, *152*, **357**, *357*
—._crystals_number, 97, **233**
—.details, 81, 151, **357**, *357*
—.entry_id, 151, *152*, **357**, *357*
—.method, *81*, 151, *152*, **357**, *357*
—.method_details, 81, 151, *152*, *357*, **358**
—._special_details, 97, 98, **233**
EXPTL_CRYSTAL, 76, 97, 98, 134, 151, 152, **233**, **286**, 342, **358**, 359–362, 408
—.colour, *38*, 151, *152*, **358**, *358*
—._colour, *6*, 76, 90, 97, 98, 118, **233**, *233*, *563*, 565
—.colour_lustre, 151, **358**
—._colour_lustre, 97, 98, **233**
—.colour_modifier, 151, **358**
—._colour_modifier, 97, **233**, 234
—.colour_primary, 151, **358**
—._colour_primary, 97, 233, **234**
—.density_diffrn, 151, 334, **358**, *358*
—._density_diffrn, *6*, 97, *233*, **234**, *563*, 565, 566
—.density_Matthews, 151, **358**, *358*
—.density_meas, 151, **358**, *358*, 359
—._density_meas, *6*, 76, 97, *233*, **234**, 235, *563*, 565, 566
—.density_meas_esd, **358**
—.density_meas_gt, 151, *358*, **359**
—._density_meas_gt, 97, *233*, **234**
—.density_meas_lt, 151, *358*, **359**
—._density_meas_lt, 97, *233*, **234**
—.density_meas_temp, 151, *358*, **359**
—._density_meas_temp, 97, **234**
—.density_meas_temp_esd, **359**
—.density_meas_temp_gt, 151, **359**
—._density_meas_temp_gt, 97, **234**
—.density_meas_temp_lt, 151, *358*, **359**
—._density_meas_temp_lt, 97, *233*, **234**
—.density_method, 151, *358*, **359**
—._density_method, *6*, 97, *233*, **235**, *563*, 565
—.density_percent_sol, 151, *152*, *358*, **359**
EXPTL_CRYSTAL
—.description, 151, *152*, *358*, **359**
—._description, *6*, 98, *233*, **235**, *563*, 565
—.F_000, 151, *358*, **359**
—._F_000, 98, *233*, **235**, *563*
—.id, *38*, 151, *152*, 342, *358*, **359**, 360–362, 408
—._id, 97, 98, 228, **235**, 248
—.preparation, 151, 357, *358*, **360**
—._preparation, 98, 233, **235**
—._pressure_history, 98, **235**
—._recrystallization_method, 98, **235**
—._size_length, 98, **235**
—.size_max, 151, *152*, *358*, **360**
—._size_max, *6*, 98, *233*, **235**, *563*, 565
—.size_mid, 151, *152*, *358*, **360**
—._size_mid, *6*, 98, *233*, **235**, *563*, 565
—.size_min, 151, *152*, *358*, **360**
—._size_min, *6*, 98, *233*, **235**, *563*, 565
—.size_rad, 151, *358*, **360**
—._size_rad, *6*, 98, **235**, 565
—._thermal_history, 98, **235**
—._type_of_structure, 134, *134*, *137, 138*, **286**
EXPTL_CRYSTAL_FACE, 76, 97, 98, 132, 134, 151, **235**, **286**, 359, **360**
—.crystal_id, 151, **360**, *360*
—.diffr_chi, 151, **360**, *360*
—._diffr_chi, 98, **235**
—.diffr_kappa, 151, **360**, *360*
—._diffr_kappa, 98, **235**
—.diffr_phi, 151, **360**, *360*
—._diffr_phi, 98, **235**
—.diffr_psi, 151, **360**, *360*
—._diffr_psi, 98, **235**
—.index_h, 151, **360**, *360*
—._index_h, 98, **235**, *235*
—.index_k, 151, **360**, *360*
—._index_k, 98, **235**, *235*
—.index_l, 151, **360**, *360*
—._index_l, 98, **235**, *235*
—._index_m_*n* ($n = 1, \ldots, 8$), *134*, **286**
—.perp_dist, 151, **360**, *360*
—._perp_dist, 76, 98, **235**, *235*, 286
EXPTL_CRYSTAL_GROW, 148, 151, 152, **360**
—.apparatus, 152, *152*, **360**, *360*
—.atmosphere, 152, *152*, *360*, **361**
—.crystal_id, 152, *152*, *360*, **361**
—.details, 152, *152*, **361**
—.method, 152, *152*, *360*, **361**
—.method_ref, 152, **361**
—.pH, 152, *152*, *360*, **361**
—.pressure, 152, **361**
—.pressure_esd, **361**
—.seeding, 152, **361**
—.seeding_ref, 152, **361**
—.temp, 152, *152*, *360*, **361**
—.temp_details, 152, **361**
—.temp_esd, **361**
—.time, 152, *152*, *360*, **361**
EXPTL_CRYSTAL_GROW_COMP, 148, 151, 152, **361**, 362
—.conc, 152, *152*, *361*, **362**
—.crystal_id, 152, *152*, *361*, **362**
—.details, 152, *152*, 361, *361*, **362**
—.id, 152, *152*, *361*, **362**
—.name, 152, *152*, *361*, **362**
—.sol_id, 152, *152*, *361*, **362**
—.volume, 152, *152*, *361*, **362**

GEOM, 105, 108, 109, 134, 164, 176, **236**, 356, **362**
—.details, 176, **362**
—.entry_id, 176, 195, **362**
—._special_details, 108, 109, **236**, *236*
GEOM_ANGLE, 108, 132, 134, 136, 176, **236**, **286**, **362**
—, 108, *109*, **236**, *236*, 566
—.atom_site_auth_asym_id_1, 176, **362**
—.atom_site_auth_asym_id_2, 176, **362**
—.atom_site_auth_asym_id_3, 176, **362**
—.atom_site_auth_atom_id_1, 176, **362**

GEOM_ANGLE
—.atom_site_auth_atom_id_2, 176, **362**
—.atom_site_auth_atom_id_3, 176, **363**
—.atom_site_auth_comp_id_1, 176, **363**
—.atom_site_auth_comp_id_2, 176, **363**
—.atom_site_auth_comp_id_3, 176, **363**
—.atom_site_auth_seq_id_1, 176, **363**
—.atom_site_auth_seq_id_2, 176, **363**
—.atom_site_auth_seq_id_3, 176, **363**
—.atom_site_id_1, 176, *362*, **363**
—.atom_site_id_2, 176, *362*, **363**, 364
—.atom_site_id_3, 176, *362*, **363**
—_atom_site_label_1, 108, *109*, **236**, *236*, 286, 566
—_atom_site_label_2, 108, *109*, **236**, *236*, 566
—_atom_site_label_3, 108, *109*, **236**, *236*, 566
—.atom_site_label_alt_id_1, 176, **363**
—.atom_site_label_alt_id_2, 177, **363**
—.atom_site_label_alt_id_3, 177, **363**
—.atom_site_label_asym_id_1, 176, **363**
—.atom_site_label_asym_id_2, 177, **363**
—.atom_site_label_asym_id_3, 177, **363**
—.atom_site_label_atom_id_1, 176, **363**
—.atom_site_label_atom_id_2, 177, **363**
—.atom_site_label_atom_id_3, 177, **363**
—.atom_site_label_comp_id_1, 177, **363**
—.atom_site_label_comp_id_2, 177, **363**
—.atom_site_label_comp_id_3, 177, **363**
—.atom_site_label_seq_id_1, 177, **363**
—.atom_site_label_seq_id_2, 177, **363**
—.atom_site_label_seq_id_3, 177, **364**
—_av, 136, **286**
—_max, 136, **286**
—_min, 136, **286**
—.publ_flag, 177, *362*, **364**
—_publ_flag, 108, 109, *109*, **236**, *236*, 566
—_site_ssg_symmetry_1, 136, **286**
—_site_ssg_symmetry_2, 136, **286**
—_site_ssg_symmetry_3, 136, **286**
—.site_symmetry_1, 176, *362*, **364**
—_site_symmetry_1, 108, **236**, *236*, 566
—.site_symmetry_2, 176, *362*, **364**
—_site_symmetry_2, 108, **236**, *236*, 566
—.site_symmetry_3, 176, *362*, **364**
—_site_symmetry_3, 108, **236**, *236*, 566
—.value, 177, *362*, **364**
—.value_esd, *362*, **364**
GEOM_BOND, 75, 78, 81, 108, 109, 111, 132, 134, 136, 176, 177, **236**, **286**, 362, **364**
—.atom_site_auth_asym_id_1, 177, **364**
—.atom_site_auth_asym_id_2, 177, **364**
—.atom_site_auth_atom_id_1, 177, **364**
—.atom_site_auth_atom_id_2, 177, **364**
—.atom_site_auth_comp_id_1, 177, **364**
—.atom_site_auth_comp_id_2, 177, **364**
—.atom_site_auth_seq_id_1, 177, **364**
—.atom_site_auth_seq_id_2, 177, **365**
—.atom_site_id_1, *81*, 177, *364*, **365**
—.atom_site_id_2, *81*, 177, *364*, **365**
—_atom_site_label_1, *6*, 75, *75*, 78, *78*, 108, *109*, **236**, *236*, *500*, 565
—_atom_site_label_2, *6*, *75*, 78, *78*, 108, *109*, **236**, *236*, *500*, 565
—.atom_site_label_alt_id_1, 177, **365**
—.atom_site_label_alt_id_2, 177, **365**
—.atom_site_label_asym_id_1, 177, **365**
—.atom_site_label_asym_id_2, 177, **365**
—.atom_site_label_atom_id_1, 177, **365**
—.atom_site_label_atom_id_2, 177, **365**
—.atom_site_label_comp_id_1, 177, **365**
—.atom_site_label_comp_id_2, 177, **365**
—.atom_site_label_seq_id_1, 177, **365**
—.atom_site_label_seq_id_2, 177, **365**
—.dist, *81*, 177, *364*, **365**
—.dist_esd, *81*, *364*, **365**
—_distance, *6*, 75, *75*, *78*, 108, *109*, **236**, *236*, 237, *500*, 565
—_distance_av, 136, **286**
—_distance_max, 136, **286**
GEOM_BOND
—_distance_min, 136, **286**
—.publ_flag, *81*, 177, *364*, **365**
—_publ_flag, 108, *109*, *236*, **237**, 559, 565
—_site_ssg_symmetry_1, 136, **286**
—_site_ssg_symmetry_2, 136, **286**
—.site_symmetry_1, *81*, 177, *364*, **365**
—_site_symmetry_1, *6*, 24, 85, 108, *109*, *236*, **237**, 565
—.site_symmetry_2, *81*, 177, *364*, **365**
—_site_symmetry_2, *6*, 108, *109*, *236*, **237**, 565
—.valence, 177, **365**
—_valence, 108, 111, **237**
GEOM_CONTACT, 108, 109, 132, 134, 136, 176, 177, 236, **237**, **287**, 362, **366**
—.atom_site_auth_asym_id_1, 177, **366**
—.atom_site_auth_asym_id_2, 177, **366**
—.atom_site_auth_atom_id_1, 177, **366**
—.atom_site_auth_atom_id_2, 177, **366**
—.atom_site_auth_comp_id_1, 177, **366**
—.atom_site_auth_comp_id_2, 177, **366**
—.atom_site_auth_seq_id_1, 177, **366**
—.atom_site_auth_seq_id_2, 177, **366**
—.atom_site_id_1, 177, **366**, *366*
—.atom_site_id_2, 177, **366**, *366*
—_atom_site_label_1, 108, *110*, **237**
—_atom_site_label_2, 108, *110*, **237**
—.atom_site_label_alt_id_1, 177, **366**
—.atom_site_label_alt_id_2, 177, **366**
—.atom_site_label_asym_id_1, 177, **366**
—.atom_site_label_asym_id_2, 177, **366**
—.atom_site_label_atom_id_1, 177, **366**
—.atom_site_label_atom_id_2, 177, **366**
—.atom_site_label_comp_id_1, 177, **366**
—.atom_site_label_comp_id_2, 177, **366**
—.atom_site_label_seq_id_1, 177, **367**
—.atom_site_label_seq_id_2, 177, **367**
—.dist, 177, *366*, **367**
—.dist_esd, *366*, **367**
—_distance, 108, *110*, **237**
—_distance_av, 136, **287**
—_distance_max, 136, **287**
—_distance_min, 136, **287**
—.publ_flag, 177, *366*, **367**
—_publ_flag, 108, *110*, **237**
—_site_ssg_symmetry_1, 136, **287**
—_site_ssg_symmetry_2, 136, **287**
—.site_symmetry_1, 177, *366*, **367**
—_site_symmetry_1, 108, *110*, **237**
—.site_symmetry_2, 177, *366*, **367**
—_site_symmetry_2, 108, *110*, **237**
GEOM_HBOND, 108, 109, 176, 177, 236, **237**, 362, **367**
—.angle_DHA, 177, **367**, *367*
—_angle_DHA, 108, *110*, **237**, *237*, 566
—.angle_DHA_esd, **367**
—.atom_site_auth_asym_id_A, 177, **367**
—.atom_site_auth_asym_id_D, 177, **367**
—.atom_site_auth_asym_id_H, 177, **367**
—.atom_site_auth_atom_id_A, 177, **367**
—.atom_site_auth_atom_id_D, 177, **367**
—.atom_site_auth_atom_id_H, 177, **367**
—.atom_site_auth_comp_id_A, 177, **368**
—.atom_site_auth_comp_id_D, 177, **368**
—.atom_site_auth_comp_id_H, 177, **368**
—.atom_site_auth_seq_id_A, 177, **368**
—.atom_site_auth_seq_id_D, 177, **368**
—.atom_site_auth_seq_id_H, 177, **368**
—.atom_site_id_A, 177, *367*, **368**
—.atom_site_id_D, 177, *367*, **368**
—.atom_site_id_H, 177, *367*, **368**
—_atom_site_label_A, 108, *110*, **237**, *237*, 566
—.atom_site_label_alt_id_A, 177, **368**
—.atom_site_label_alt_id_D, 178, **368**
—.atom_site_label_alt_id_H, 178, **368**
—.atom_site_label_asym_id_A, 177, **368**
—.atom_site_label_asym_id_D, 178, **368**
—.atom_site_label_asym_id_H, 178, **368**
—.atom_site_label_atom_id_A, 178, **368**
GEOM_HBOND
—.atom_site_label_atom_id_D, 178, **368**
—.atom_site_label_atom_id_H, 178, **368**
—.atom_site_label_comp_id_A, 178, **368**
—.atom_site_label_comp_id_D, 178, **368**
—.atom_site_label_comp_id_H, 178, **368**
—_atom_site_label_D, 108, *110*, **237**, *237*, 566
—_atom_site_label_H, 108, 109, *110*, **237**, *237*, 566
—.atom_site_label_seq_id_A, 178, **368**
—.atom_site_label_seq_id_D, 178, **369**
—.atom_site_label_seq_id_H, 178, **369**
—.dist_DA, 178, *367*, **369**
—.dist_DA_esd, **369**
—.dist_DH, 178, *367*, **369**
—.dist_DH_esd, **369**
—.dist_HA, 178, *367*, **369**
—.dist_HA_esd, **369**
—_distance_DA, 108, *110*, *237*, **238**, 566
—_distance_DH, 108, *110*, *237*, **238**, 566
—_distance_HA, 108, *110*, *237*, **238**, 566
—.publ_flag, 178, *367*, **369**
—_publ_flag, 108, *110*, *237*, **238**, 566
—.site_symmetry_A, 177, **369**
—_site_symmetry_A, 108, **238**, 566
—.site_symmetry_D, 177, **369**
—_site_symmetry_D, 108, **238**, 566
—.site_symmetry_H, 177, **369**
—_site_symmetry_H, 108, **238**, 566
GEOM_TORSION, 108–110, 132, 134, 136, 176, 178, 236, **238**, **287**, 362, **369**
—, 109, *110*, **238**, 566
—.atom_site_auth_asym_id_1, 178, **370**
—.atom_site_auth_asym_id_2, 178, **370**
—.atom_site_auth_asym_id_3, 178, **370**
—.atom_site_auth_asym_id_4, 178, **370**
—.atom_site_auth_atom_id_1, 178, **370**
—.atom_site_auth_atom_id_2, 178, **370**
—.atom_site_auth_atom_id_3, 178, **370**
—.atom_site_auth_atom_id_4, 178, **370**
—.atom_site_auth_comp_id_1, 178, **370**
—.atom_site_auth_comp_id_2, 178, **370**
—.atom_site_auth_comp_id_3, 178, **370**
—.atom_site_auth_comp_id_4, 178, **370**
—.atom_site_auth_seq_id_1, 178, **370**
—.atom_site_auth_seq_id_2, 178, **370**
—.atom_site_auth_seq_id_3, 178, **370**
—.atom_site_auth_seq_id_4, 178, **370**
—.atom_site_id_1, 178, *369*, **370**
—.atom_site_id_2, 178, 369, *369*, **370**
—.atom_site_id_3, 178, *369*, **370**
—.atom_site_id_4, 178, *369*, **370**
—_atom_site_label_1, 109, *110*, **238**, 566
—_atom_site_label_2, 109, *110*, **238**, 566
—_atom_site_label_3, 109, *110*, **238**, 566
—_atom_site_label_4, 109, *110*, **238**, 566
—.atom_site_label_alt_id_1, 178, **370**
—.atom_site_label_alt_id_2, 178, **370**
—.atom_site_label_alt_id_3, 178, **371**
—.atom_site_label_alt_id_4, 178, **371**
—.atom_site_label_asym_id_1, 178, **371**
—.atom_site_label_asym_id_2, 178, **371**
—.atom_site_label_asym_id_3, 178, **371**
—.atom_site_label_asym_id_4, 178, **371**
—.atom_site_label_atom_id_1, 178, **371**
—.atom_site_label_atom_id_2, 178, **371**
—.atom_site_label_atom_id_3, 178, **371**
—.atom_site_label_atom_id_4, 178, **371**
—.atom_site_label_comp_id_1, 178, **371**
—.atom_site_label_comp_id_2, 178, **371**
—.atom_site_label_comp_id_3, 178, **371**
—.atom_site_label_comp_id_4, 178, **371**
—.atom_site_label_seq_id_1, 178, **371**
—.atom_site_label_seq_id_2, 178, **371**
—.atom_site_label_seq_id_3, 178, **371**
—.atom_site_label_seq_id_4, 178, **371**
—_av, 136, **287**
—_max, 136, **287**
—_min, 136, **287**

GEOM_TORSION
—.publ_flag, 178, *369*, **371**
—._publ_flag, 109, *110*, **238**, 566
—._site_ssg_symmetry_1, 136, **287**
—._site_ssg_symmetry_2, 136, **287**
—._site_ssg_symmetry_3, 136, **287**
—._site_ssg_symmetry_4, 136, **287**
—.site_symmetry_1, 178, *369*, **371**
—._site_symmetry_1, 109, *110*, **238**, 566
—.site_symmetry_2, 178, *369*, **371**
—._site_symmetry_2, 109, *110*, **238**, 566
—.site_symmetry_3, 178, *369*, **372**
—._site_symmetry_3, 109, *110*, **238**, 566
—.site_symmetry_4, 178, *369*, **372**
—._site_symmetry_4, 109, *110*, **238**, 566
—.value, 178, *369*, **372**
—.value_esd, **372**

_item.category_id, *see* DDL2
_item.mandatory_code, *see* DDL2
_item.name, *see* DDL2
_item_aliases.alias_name, *see* DDL2
_item_aliases.dictionary, *see* DDL2
_item_aliases.name, *see* DDL2
_item_aliases.version, *see* DDL2
_item_default.name, *see* DDL2
_item_default.value, *see* DDL2
_item_dependent.dependent_name, *see* DDL2
_item_dependent.name, *see* DDL2
_item_description.description, *see* DDL2
_item_description.name, *see* DDL2
_item_enumeration.detail, *see* DDL2
_item_enumeration.name, *see* DDL2
_item_enumeration.value, *see* DDL2
_item_examples.case, *see* DDL2
_item_examples.detail, *see* DDL2
_item_linked.child_name, *see* DDL2
_item_linked.parent_name, *see* DDL2
_item_range.maximum, *see* DDL2
_item_range.minimum, *see* DDL2
_item_related.function_code, *see* DDL2
_item_related.name, *see* DDL2
_item_related.related_name, *see* DDL2
_item_structure.code, *see* DDL2
_item_structure.organization, *see* DDL2
_item_structure_list.code, *see* DDL2
_item_structure_list.dimension, *see* DDL2
_item_structure_list.index, *see* DDL2
_item_sub_category.id, *see* DDL2
_item_type.code, *see* DDL2
_item_type.name, *see* DDL2
_item_type_conditions.code, *see* DDL2
_item_type_list.code, *see* DDL2
_item_type_list.construct, *see* DDL2
_item_type_list.detail, *see* DDL2
_item_type_list.primitive_code, *see* DDL2
_item_units.code, *see* DDL2
_item_units.name, *see* DDL2
_item_units_conversion.factor, *see* DDL2
_item_units_conversion.from_code, *see* DDL2
_item_units_conversion.operator, *see* DDL2
_item_units_conversion.to_code, *see* DDL2
_item_units_list.code, *see* DDL2
_item_units_list.detail, *see* DDL2
ITEM, 63, 65–68, **474**
ITEM_ALIASES, 63, 66, **474**
ITEM_DEFAULT, 66, **475**
ITEM_DEPENDENT, 64, 66, **475**
ITEM_DESCRIPTION, 63, 66, **475**
ITEM_ENUMERATION, 67, **475**
ITEM_EXAMPLES, 64, 67, **475**
ITEM_LINKED, 65–67, **475**
ITEM_METHODS, 67, **475**
ITEM_RANGE, 64, 67, **475**
ITEM_RELATED, 64, 67, **475**
ITEM_STRUCTURE, 67, **476**
ITEM_STRUCTURE_LIST, 67, **476**
ITEM_SUB_CATEGORY, 64, 67, **476**
ITEM_TYPE, 63, 67–69, **476**
ITEM_TYPE_CONDITIONS, 68, **476**
ITEM_TYPE_LIST, 63, 64, 67, 68, **476**
ITEM_UNITS, 64, 68, **477**
ITEM_UNITS_CONVERSION, 64, 68, **477**
ITEM_UNITS_LIST, 64, 68, **477**

JOURNAL, 84, 113, 193, 194, **239**, 314, **372**
—.coden_ASTM, 193, **372**, *372*
—._coden_ASTM, 113, **239**, *239*
—.coden_Cambridge, 193, **372**
—._coden_Cambridge, 113, **239**
—.coeditor_address, 193, **372**
—._coeditor_address, 113, **239**
—.coeditor_code, 193, **372**, *372*
—._coeditor_code, 113, **239**, *239*
—.coeditor_email, 193, **372**
—._coeditor_email, 113, **239**
—.coeditor_fax, 193, **372**
—._coeditor_fax, 113, **239**
—.coeditor_name, 193, **372**
—._coeditor_name, 113, **239**
—.coeditor_notes, 193, **372**
—._coeditor_notes, 113, **239**
—.coeditor_phone, 193, **372**
—._coeditor_phone, 84, 113, **239**
—.data_validation_number, 193, **372**
—._data_validation_number, 113, **239**
—.date_accepted, 193, **372**, *372*
—._date_accepted, 113, **239**, *239*
—.date_from_coeditor, 193, **372**, *372*
—._date_from_coeditor, 113, **239**, *239*
—.date_printers_final, 193, **372**
—._date_printers_final, 113, **239**
—.date_printers_first, 193, **372**, *372*
—._date_printers_first, 113, **239**, *239*
—.date_proofs_in, 193, **372**
—._date_proofs_in, 113, **239**
—.date_proofs_out, 193, **372**, *372*
—._date_proofs_out, 113, **239**, *239*
—.date_recd_copyright, 193, **373**
—._date_recd_copyright, 113, **239**
—.date_recd_electronic, 193, *372*, **373**
—._date_recd_electronic, 113, **239**, *239*
—.date_recd_hard_copy, 193, **373**
—._date_recd_hard_copy, 113, **239**
—.date_to_coeditor, 193, **373**
—._date_to_coeditor, 113, **239**
—.entry_id, 193, *372*, **373**
—.issue, 193, *372*, **373**
—._issue, 113, **239**, *239*
—.language, 193, **373**
—._language, 113, **239**
—.name_full, 193, *372*, **373**
—._name_full, 113, **239**, *239*
—.page_first, 193, *372*, **373**
—._page_first, 113, **239**, *239*
—.page_last, 193, *372*, **373**
—._page_last, 113, **239**, *239*
—.paper_category, 193, **373**
—._paper_category, 113, **239**
—.suppl_publ_number, 193, **373**
—._suppl_publ_number, 113, **239**
—.suppl_publ_pages, 193, **373**
—._suppl_publ_pages, 113, **239**
—.techeditor_address, 193, **373**
—._techeditor_address, 113, **239**
—.techeditor_code, 193, *372*, **373**
—._techeditor_code, 113, **239**, *239*
—.techeditor_email, 193, **373**
—._techeditor_email, 113, **239**
—.techeditor_fax, 193, **373**
—._techeditor_fax, 113, **239**
—.techeditor_name, 193, **373**
—._techeditor_name, 113, **239**
—.techeditor_notes, 193, **373**
—._techeditor_notes, 113, **239**
—.techeditor_phone, 193, **373**
—._techeditor_phone, 84, 113, **239**
—.volume, 193, *372*, **373**
JOURNAL
—._volume, 113, **239**, *239*
—.year, 193, *372*, **373**
—._year, 113, **239**, *239*
JOURNAL_INDEX, 113, 193, 194, **239**, **373**
—.subterm, 193, **373**, *373*
—._subterm, 113, *114*, **239**, *239*
—.term, 193, **373**, *373*
—._term, 113, *114*, **239**, *239*
—.type, 193, *373*, **374**
—._type, 113, *114*, **239**, *239*

_list, *see* DDL1
_list_level, *see* DDL1
_list_link_child, *see* DDL1
_list_link_parent, *see* DDL1
_list_mandatory, *see* DDL1
_list_reference, *see* DDL1
_list_uniqueness, *see* DDL1

_method_list.code, *see* DDL2
_method_list.detail, *see* DDL2
_method_list.id, *see* DDL2
_method_list.inline, *see* DDL2
_method_list.language, *see* DDL2
METHOD_LIST, 67, 69, 70, **477**
MIF (Molecular Information File)
_atom_attach_all, *51*, **467**
_atom_attach_h, *46*, 47, *47*, 48, *48–51*, **467**
_atom_attach_nh, 48, *49*, **467**
_atom_attach_ring, **467**
_atom_charge, 48, *49*, **467**, 468
_atom_cip, 48, *48*, *50, 51*, **467**
_atom_coord_x, *46*, 47, *47*, **467**
_atom_coord_y, *46*, 47, *47*, **467**
_atom_coord_z, *46*, 47, *47*, **467**
_atom_environment, 48, *49*, **467**
_atom_frag_id, 48, *49*, **467**
_atom_frag_key, 48, *49*, **467**
_atom_id, *46*, 47, *47*, 48, *48–51*, **467**, 468, 470
_atom_label, *46*, 47, **467**
_atom_mass_number, *45*, **467**
_atom_radical_count, **467**, 468
_atom_spin_multiplicity, **467**
_atom_type, *45, 46*, 47, *47–51*, **467**
_atom_valency, **468**
_bond_cip, 48, *48*, **468**
_bond_environment, **468**
_bond_id_1, *46–49*, *51*, **468**
_bond_id_2, *46–49*, *51*, **468**
_bond_type_casreg3, 48, *48*, **468**
_bond_type_ccdc, 48, *48*, *51*, **468**
_bond_type_mif, *46*, 47, *47*, 48, *48, 49*, *51*, **468**
_define_bonding_convention, 467, **468**
_define_stereo_relationship, *48*, 50, *51*, **468**
_display_colour, *45, 46*, 47, **468**
_display_conn_colour, *46*, 48, **469**
_display_conn_id, *46*, 47, **469**
_display_conn_symbol, *46*, 48, **469**
_display_coord_x, *46*, **469**
_display_coord_y, *46*, **469**
_display_define_origin, 47, **469**
_display_define_scale, *46*, **469**
_display_define_span_x, *46*, **470**
_display_define_span_y, *46*, **470**
_display_id, 45, *46*, 48, 469, **470**
_display_object, *46*, 47, 469, **470**
_display_size, *46*, 47, **470**
_display_symbol, *46*, 468, 469, **470**
_molecule_name_cas_10ci, **470**
_molecule_name_cas_8ci, **470**
_molecule_name_cas_9ci, **470**
_molecule_name_common, *47–51*, **470**
_molecule_name_iupac, *50, 51*, **470**
_reference_conformation, *47*, **470**
_reference_stereo_group, 468, **470**
_stereo_atom_id, *48*, 50, *51*, 468, **470**
_stereo_bond_id_1, *48*, 50, 468, **470**
_stereo_bond_id_2, *48*, 50, 468, **470**

MIF (Molecular Information File)
_stereo_geometry, *48*, 50, *50, 51*, 468, **470**
_stereo_vertex_id, *48*, 50, *51*, 468, **470**
_molecule_name_cas_10ci, *see* MIF
_molecule_name_cas_8ci, *see* MIF
_molecule_name_cas_9ci, *see* MIF
_molecule_name_common, *see* MIF
_molecule_name_iupac, *see* MIF

_name, *see* DDL1

PD_BLOCK, 123, **258**, 282
—_diffractogram_id, *124*
—_id, 123, 124, *124*, 125, *125*, **258**, 259, 263, 266, 282
PD_CALC, 121, 122, **258**
—_method, 122, *125*, **259**
PD_CALIB, 118, 119, **259**
—_detector_id, 119, 127, **259**, 261
—_detector_response, 119, 127, **259**
—_std_external_block_id, 119, 124, **259**, *259*
—_std_external_name, 119, **259**, *259*
—_std_internal_mass_%, 119, **259**
—_std_internal_name, 119, **259**
—_2theta_off_max, 119, **259**
—_2theta_off_min, 119, **259**
—_2theta_off_point, 119, **259**
—_2theta_offset, 119, 127, *127*, **259**
PD_CALIBRATION, 118, 119
—_conversion_eqn, 119, 127, **259**, *259*, 263
—_special_details, 119, **260**
PD_CHAR, 118, 119, **260**
—_atten_coef_mu_calc, 118, **260**
—_atten_coef_mu_obs, 118, **260**
—_colour, 118, **260**
—_particle_morphology, 118, **260**
—_special_details, 118, **260**
PD_DATA, 77, 118–122, **260**
_pd_calc_intensity_net, 122, 128, **259**
_pd_calc_intensity_total, 122, *125*, 128, **259**, *260*, 268
_pd_calc_point_id, 122, **259**, *260*
_pd_instr_beam_size_ax, 119, 120, **261**
_pd_instr_beam_size_eq, 119, 120, **261**
_pd_instr_var_illum_len, 119, 120, 261, **263**
_pd_meas_angle_chi, 120, 127, **263**
_pd_meas_angle_omega, 120, 127, **263**
_pd_meas_angle_phi, 120, 127, **263**
_pd_meas_angle_2theta, 120, **263**
_pd_meas_counts_background, 120, 127, **263**
_pd_meas_counts_container, 120, 127, **263**
_pd_meas_counts_monitor, 120, 126, 128, **263**
_pd_meas_counts_total, 120, 126–128, **263**
_pd_meas_detector_id, 120, 126, 127, 259, **263**, 264
_pd_meas_intensity_background, 121, **264**
_pd_meas_intensity_container, 121, **264**
_pd_meas_intensity_monitor, 121, 128, **264**
_pd_meas_intensity_total, 121, *125*, 126–128, *260*, **264**, 268
_pd_meas_point_id, 120, 121, *125*, *260*, **264**, 268
_pd_meas_position, 120, 127, 259, **264**
_pd_meas_rocking_angle, 120, 128, **264**
_pd_meas_step_count_time, 120, *126*, 263, *263*, **264**, 265
_pd_meas_time_of_flight, 120, 125, 127, 128, **265**
_pd_meas_2theta_scan, 120, 126–128, 264, **265**
_pd_proc_d_spacing, 121, 122, *125*, 127, 128, **266**
_pd_proc_energy_detection, 121, 127, **266**
_pd_proc_energy_incident, 121, **266**
_pd_proc_intensity_bkg_calc, 121, 122, *125*, *260*, **267**
_pd_proc_intensity_bkg_fix, 121, 122, **267**
_pd_proc_intensity_incident, 121, 122, 127, **267**
_pd_proc_intensity_net, 121, 126–128, 259, **267**, 268
PD_DATA
_pd_proc_intensity_norm, 121, 122, **267**
_pd_proc_intensity_total, 121, *125*, 128, **267**, 268
_pd_proc_ls_weight, 121, *125*, 128, *260*, **268**
_pd_proc_point_id, 121, *125*, *260*, **268**
_pd_proc_recip_len_Q, 121, 127, 128, **268**
_pd_proc_2theta_corrected, 121, 122, 127, 128, 259, 265, **267**
_pd_proc_wavelength, 121, 122, 267, **268**
—_point_id, 118, 120, 121, 259, **260**, *260*, 264, 268
PD_INSTR, 118, 119, **261**
—_cons_illum_flag, 119, 120, *120*, **261**, *261*
—_cons_illum_len, 119, *120*, **261**, 263
—_dist_anal/detc, 119, **261**
—_dist_mono/spec, 119, **261**
—_dist_spec/anal, 119, **261**
—_dist_spec/detc, 119, **261**
—_dist_src/mono, 119, **261**
—_dist_src/spec, 119, **261**
—_divg_ax_anal/detc, 119, **261**
—_divg_ax_mono/spec, 119, **261**
—_divg_ax_spec/anal, 119, **261**
—_divg_ax_spec/detc, 119, **261**
—_divg_ax_src/mono, 119, **261**
—_divg_ax_src/spec, 119, **261**
—_divg_eq_anal/detc, 119, **261**
—_divg_eq_mono/spec, 119, **261**
—_divg_eq_spec/anal, 119, **261**
—_divg_eq_spec/detc, 119, **261**
—_divg_eq_src/mono, 119, **261**
—_divg_eq_src/spec, 119, **261**
—_geometry, 119, 126, *261*, **262**
—_location, 119, **262**
—_monochr_post_spec, 119, *261*, **262**, 263
—_monochr_pre_spec, 119, 120, **262**, 263
—_slit_ax_anal/detc, 119, **262**
—_slit_ax_mono/spec, 119, **262**
—_slit_ax_spec/anal, 119, **262**
—_slit_ax_spec/detc, 119, **262**
—_slit_ax_src/mono, 119, **262**
—_slit_ax_src/spec, 119, **262**
—_slit_eq_anal/detc, 119, *261*, **262**
—_slit_eq_mono/spec, 119, **262**
—_slit_eq_spec/anal, 119, **262**
—_slit_eq_spec/detc, 119, **262**
—_slit_eq_src/mono, 119, **262**
—_slit_eq_src/spec, 119, *261*, **262**
—_soller_ax_anal/detc, 119, **262**
—_soller_ax_mono/spec, 119, **262**
—_soller_ax_spec/anal, 119, **262**
—_soller_ax_spec/detc, 119, **262**
—_soller_ax_src/mono, 119, **262**
—_soller_ax_src/spec, 119, **262**
—_soller_eq_anal/detc, 119, **262**
—_soller_eq_mono/spec, 119, **262**
—_soller_eq_spec/anal, 119, **262**
—_soller_eq_spec/detc, 119, **262**
—_soller_eq_src/mono, 119, **262**
—_soller_eq_src/spec, 119, **262**
—_source_size_ax, 119, 120, **263**
—_source_size_eq, 119, 120, **263**
—_special_details, 119, **263**
—_2theta_monochr_post, 119, **263**
—_2theta_monochr_pre, 119, 120, **263**
PD_MEAS, 118, **263**
PD_MEAS_INFO, 118, 121
—_author_address, 121, **263**, *263*
—_author_email, 121, **263**, *263*
—_author_fax, 121, **264**
—_author_name, 121, 263, *263*, **264**
—_author_phone, 121, **264**
PD_MEAS_METHOD, 118, 121
_pd_meas_datetime_initiated, 120, 121, 258, **263**, *263*
_pd_meas_number_of_points, 121, *125, 126*, **264**
_pd_meas_rocking_axis, 121, 127, **264**
PD_MEAS_METHOD
_pd_meas_scan_method, 121, *126, 127*, *263*, **264**, 265
_pd_meas_special_details, 121, **264**
_pd_meas_2theta_fixed, 120, *125*, 127, *127*, **264**, 265
_pd_meas_2theta_range_inc, 120, *126*, *263*, **265**
_pd_meas_2theta_range_max, 120, *126*, *263*, **265**
_pd_meas_2theta_range_min, 120, *126*, *263*, **265**
_pd_meas_units_of_intensity, 121, 126, 264, **265**
PD_PEAK, 121, 122, **265**
—_d_spacing, 122, **265**
—_id, 122, 123, 129, **265**, 268
—_intensity, 122, **265**
—_pk_height, 122, **265**
—_2theta_centroid, 122, 129, **265**
—_2theta_maximum, 122, **265**
—_wavelength_id, 122, 130, **265**
—_width_d_spacing, 122, **265**
—_width_2theta, 122, **265**
PD_PEAK_METHOD, 121, 122
_pd_peak_special_details, 122, **265**
PD_PHASE, 123, **265**
—_block_id, 123, 124, *125*, 129, 258, 265, **266**
—_id, 123, *125*, 129, 265, **266**, 268
—_mass_%, 123, *125*
—_name, 123, *124*, 129, **266**
PD_PREP, 118, 119, **266**
—_conditions, 118, **266**
—_cool_rate, 118, **266**
—_pressure, 118, **266**
—_temperature, 118, **266**
PD_PROC, 121, **266**
_pd_block_diffractogram_id, 123, 124, 129, **258**
PD_PROC_INFO, 121
_pd_proc_number_of_points, *125*, 259, **268**
_pd_proc_2theta_range_inc, 121, **267**
_pd_proc_2theta_range_max, 121, **267**
_pd_proc_2theta_range_min, 121, **267**
—_author_address, 121, **266**
—_author_email, 121, **266**
—_author_fax, 121, **266**
—_author_name, 121, **266**
—_author_phone, 121, **266**
—_data_reduction, 121, 122, **266**
—_datetime, 121, *125*, 258, **266**
—_excluded_regions, 121, 122, **267**, 268
—_special_details, 121, **267**
PD_PROC_LS, 121, **267**
—_background_function, 121, 122, **267**
—_peak_cutoff, 121, *125*, **267**
—_pref_orient_corr, 121, **267**
—_prof_R_factor, 121, **268**
—_prof_wR_expected, 121, *125*, **268**
—_prof_wR_factor, 121, *124, 125*, **268**
—_profile_function, 121, 129, **268**
—_special_details, 121, **268**
PD_REFLN, **268**
PD_SPEC, 118, 119, **269**
—_description, 119, **269**
—_mount_mode, 119, **269**, *269*
—_mounting, 119, **269**, *269*
—_orientation, 119, **269**, *269*
—_preparation, 119, 260, **269**, *269*
—_shape, 119, **269**
—_size_axial, 119, **269**
—_size_equat, 119, **269**
—_size_thick, 119, **269**
—_special_details, 119, **269**
PHASING, 152, 153, 167, **374**
—.method, 153, *153*, **374**, *374*
PHASING_AVERAGING, 152, 153, **374**
—.details, 153, *153*, **374**, *374*
—.entry_id, 153, *153*, **374**, *374*
—.method, 153, *153*, **374**, *374*

PHASING_ISOMORPHOUS, 152–154, **374**
—.details, 153, *154*, **374**, *374*
—.entry_id, 153, *154*, **374**
—.method, 153, **374**
—.parent, 153, *154*, **374**, *374*
PHASING_MAD, 152, 154, 155, 157, **374**
—.details, 154, **374**
—.entry_id, 154, *154*, **374**, *374*
—.method, 154, **375**
PHASING_MAD_CLUST, 152, 154, 155, **375**, 376, 377
—.expt_id, 154, *154*, **375**, *375*
—.id, 154, *154*, **375**, *375*, 376, 377
—.number_set, 154, *154*, **375**, *375*
PHASING_MAD_EXPT, 153–155, **375**, 376, 377
—.delta_delta_phi, 154, *154*, **375**, *375*
—.delta_phi, 154, **375**
—.delta_phi_sigma, 154, *154*, **375**, *375*
—.id, 154, *154*, **375**, *375*, 376, 377
—.mean_fom, 154, *154*, **375**, *375*
—.number_clust, 154, *154*, **375**, *375*
—.R_normal_all, 154, *154*, **375**, *375*
—.R_normal_anom_scat, 154, *154*, **375**, *375*
PHASING_MAD_RATIO, 153–155, **375**
—.clust_id, 154, *154*, **376**, *376*
—.d_res_high, 154, *154*, 155, **376**, *376*
—.d_res_low, 154, *154*, 155, **376**, *376*
—.expt_id, 154, *154*, **376**, *376*
—.ratio_one_wl, 154, *154*, **376**, *376*
—.ratio_one_wl_centric, 154, *154*, **376**, *376*
—.ratio_two_wl, 154, *154*, **376**, *376*
—.wavelength_1, 154, *154*, **376**, *376*
—.wavelength_2, 154, *154*, **376**, *376*
PHASING_MAD_SET, 153–155, 376, **377**
—.clust_id, 154, *154*, **377**, *377*
—.d_res_high, 154, *154*, **377**, *377*
—.d_res_low, 154, *154*, **377**, *377*
—.expt_id, 154, *154*, **377**, *377*
—.f_double_prime, 154, *154*, **377**, *377*
—.f_prime, 154, *154*, **377**, *377*
—.set_id, 154, *154*, **377**, *377*
—.wavelength, 154, *154*, 376, **377**, *377*
—.wavelength_details, 154, *154*, **377**, *377*
PHASING_MIR, 153–157, **377**
—.d_res_high, 155, **377**
—.d_res_low, 155, **377**
—.details, 155, **377**
—.entry_id, 155, *157*, **377**
—.FOM, 155, **378**
—.FOM_acentric, 155, **378**
—.FOM_centric, 155, **378**
—.method, 155, *157*, *377*, **378**
—.reflns, 155, **378**
—.reflns_acentric, 155, **378**
—.reflns_centric, 155, **378**
—.reflns_criterion, 155, **378**
PHASING_MIR_DER, 153, 155, 156, **378**, 380–383
—.d_res_high, 155, **378**
—.d_res_low, 155, **378**
—.der_set_id, 155, 156, **378**
—.details, 155, *157*, **378**, *378*
—.id, 155, *157*, **378**, *378*, 380, 381, 383
—.native_set_id, 155, 156, **378**
—.number_of_sites, 155, *157*, **378**, *378*
—.power_acentric, 155, **379**
—.power_centric, 155, **379**
—.R_cullis_acentric, 155, **379**
—.R_cullis_anomalous, 155, **379**
—.R_cullis_centric, 155, **379**
—.reflns_acentric, 155, **379**
—.reflns_anomalous, 155, **379**
—.reflns_centric, 155, **379**
—.reflns_criteria, 155, **379**
PHASING_MIR_DER_REFLN, 153, 155, 156, **380**
—.der_id, 155, 156, *157*, **380**, *380*
—.F_calc, 155, **380**
—.F_calc_au, 155, *157*, **380**, *380*
PHASING_MIR_DER_REFLN
—.F_meas, 155, **380**
—.F_meas_au, 155, *157*, **380**, *380*
—.F_meas_sigma, 155, *157*, **380**, *380*
—.F_meas_sigma_au, 155, **380**
—.HL_A_iso, 155, *157*, **380**, *380*
—.HL_B_iso, 155, *157*, **380**, *380*
—.HL_C_iso, 155, *157*, *380*, **381**
—.HL_D_iso, 156, *157*, *380*, **381**
—.index_h, 155, *157*, *380*, **381**
—.index_k, 155, *157*, *380*, **381**
—.index_l, 155, *157*, *380*, **381**
—.phase_calc, 156, *157*, *380*, **381**
—.set_id, 155, 156, *157*, *380*, **381**
PHASING_MIR_DER_SHELL, 153, 156, **381**
—.d_res_high, 156, *157*, **381**, *381*
—.d_res_low, 156, *157*, **381**, *381*
—.der_id, 156, *157*, **381**, *381*
—.fom, 156, **381**
—.ha_ampl, 156, *157*, *381*, **382**
—.loc, 156, *157*, *381*, **382**
—.phase, 156, **382**
—.power, 156, **382**
—.R_cullis, 156, **382**
—.R_kraut, 156, **382**
—.reflns, 156, **382**
PHASING_MIR_DER_SITE, 153, 156, **382**, 383
—.atom_type_symbol, 156, *157*, **382**, *382*
—.B_iso, 156, *157*, **382**, *382*
—.B_iso_esd, **382**
—.Cartn_x, 156, **382**, 383
—.Cartn_x_esd, **383**
—.Cartn_y, 156, **383**
—.Cartn_y_esd, **383**
—.Cartn_z, 156, **383**
—.Cartn_z_esd, **383**
—.der_id, 156, *157*, 382, *382*, **383**
—.details, 156, **383**
—.fract_x, 156, *157*, *382*, **383**
—.fract_x_esd, **383**
—.fract_y, 156, *157*, *382*, **383**
—.fract_y_esd, **383**
—.fract_z, 156, *157*, *382*, **383**
—.fract_z_esd, **383**
—.id, 156, *157*, *382*, **383**
—.occupancy, 156, *157*, *382*, **383**
—.occupancy_anom, 156, **383**
—.occupancy_anom_su, 156, **383**
—.occupancy_iso, 156, **383**
—.occupancy_iso_su, 156, **383**
PHASING_MIR_SHELL, 155, 156, **384**
—.d_res_high, 155, *157*, **384**, *384*
—.d_res_low, 155, *157*, **384**, *384*
—.FOM, 155, **384**, *384*
—.FOM_acentric, 155, **384**
—.FOM_centric, 155, **384**
—.loc, 155, **384**
—.mean_phase, 155, **384**
—.power, 155, **384**
—.R_cullis, 155, **384**
—.R_kraut, 155, **384**
—.reflns, 155, *157*, *384*, **385**
—.reflns_acentric, 155, **385**
—.reflns_anomalous, 155, **385**
—.reflns_centric, 155, **385**
PHASING_SET, 153, 155, 157, 377, 378, 381, **385**, 386
—.cell_angle_alpha, 157, *157*, **385**, *385*
—.cell_angle_beta, 157, *157*, **385**, *385*
—.cell_angle_gamma, 157, *157*, **385**, *385*
—.cell_length_a, 157, *157*, **385**, *385*
—.cell_length_b, 157, *157*, **385**, *385*
—.cell_length_c, 157, *157*, **385**, *385*
—.detector_specific, 157, *157*, **385**, *385*
—.detector_type, 157, *157*, **385**, *385*
—.id, 157, *157*, 377, 378, 381, **385**, *385*, 386
—.radiation_source_specific, 157, **385**
—.radiation_wavelength, 157, *157*, **385**, *385*
—.temp, 157, **385**
PHASING_SET_REFLN, 153, 155–158, 380, **386**
—.F_meas, 157, **386**
—.F_meas_au, 157, *157*, **386**, *386*
—.F_meas_sigma, 157, **386**
—.F_meas_sigma_au, 157, *157*, **386**, *386*
—.index_h, 157, *157*, **386**, *386*
—.index_k, 157, *157*, **386**, *386*
—.index_l, 157, *157*, **386**, *386*
—.set_id, 157, *157*, 158, **386**, *386*
PUBL, 114, 145, 190, 193, 194, **239**, **386**
—.contact_author, 194, **386**
—._contact_author, 114, **239**
—.contact_author_address, 194, **386**
—._contact_author_address, 114, **240**, 565
—.contact_author_email, 194, **386**
—._contact_author_email, 114, **240**, 565
—.contact_author_fax, 194, **387**
—._contact_author_fax, 114, **240**, 565
—._contact_author_id_iucr, 114, **240**
—.contact_author_name, 194, 386, **387**
—._contact_author_name, 114, 239, **240**, 565
—.contact_author_phone, 194, **387**
—._contact_author_phone, 114, **240**, 565
—.contact_letter, 194, **387**
—._contact_letter, 114, **240**, 565
—.entry_id, 194, **387**
—.manuscript_creation, 194, **387**
—._manuscript_creation, 24, 114, **240**
—.manuscript_processed, 194, **387**, 388
—._manuscript_processed, 24, 114, **240**
—.manuscript_text, 194, **387**, 388
—._manuscript_text, 114, **240**
—.requested_category, 194, **387**
—._requested_category, 114, **240**, 565
—.requested_coeditor_name, 194, **387**
—._requested_coeditor_name, 114, **240**
—.requested_journal, 194, **387**
—._requested_journal, 114, **240**, 565
—.section_abstract, 194, *386*, **387**
—._section_abstract, 114, *239*, **240**, *559*, 565
—.section_acknowledgements, 194, **387**
—._section_acknowledgements, 114, **240**, 565
—.section_comment, 194, **387**
—._section_comment, 114, **240**, 559, 565
—.section_discussion, 194, **387**
—._section_discussion, 114, **240**
—.section_experimental, 194, **388**
—._section_experimental, *6*, 114, **240**
—.section_exptl_prep, 194, **388**
—._section_exptl_prep, 114, **240**, 565
—.section_exptl_refinement, 194, **388**
—._section_exptl_refinement, 114, **240**, *561*, 565
—.section_exptl_solution, 194, **388**
—._section_exptl_solution, 98, 114, **240**
—.section_figure_captions, 194, **388**
—._section_figure_captions, 114, **240**, 565
—.section_introduction, 194, **388**
—._section_introduction, 114, **240**
—.section_references, 191, 194, **388**
—._section_references, 83, 84, 112, 114, 191, **240**, 565
—.section_synopsis, 194, **388**
—._section_synopsis, 114, **240**, 565
—.section_table_legends, 194, **388**
—._section_table_legends, 114, **240**
—.section_title, 194, **388**
—._section_title, 114, **240**, *559*, 565
—.section_title_footnote, 194, *386*, **388**
—._section_title_footnote, 114, *239*, **240**, 565
PUBL_AUTHOR, 114, 115, 193, 194, **241**, 388, **388**
—.address, 194, **388**, *388*, 389
—._address, 114, **241**, *241*, *559*, 565
—.email, 194, **388**
—._email, **241**
—.footnote, 194, **388**
—._footnote, 114, **241**, 565
—.id_iucr, 194, **389**
—._id_iucr, 114, **241**

PUBL_AUTHOR
—.name, 194, 388, *388*, **389**
—_name, 114, **241**, *241*, *559*, 565
PUBL_BODY, 114, 193, 194, 239, **241**, **389**
—.contents, 194, **389**, *389*
—_contents, 114, **241**, *241*
—.element, 194, **389**, *389*
—_element, 114, **241**, *241*, 242
—.format, 194, **389**, *389*
—_format, 114, *241*, **242**
—.label, 194, **389**, *389*
—_label, 114, *241*, **242**
—.title, 194, **389**, *389*
—_title, 114, *241*, **242**
PUBL_MANUSCRIPT_INCL, 114, 193, 194, **242**, **389**, 558
—.entry_id, 194, **389**, *389*
—.extra_defn, 194, *389*, **390**
—_extra_defn, 114, **242**, *242*, 559, *560*, 566
—.extra_info, 194, *389*, **390**
—_extra_info, 114, *114*, **242**, *242*
—.extra_item, 194, *389*, **390**
—_extra_item, 114, *114*, **242**, *242*, 559, *560*, 562, 566

_reference_conformation, *see* MIF
_reference_stereo_group, *see* MIF
REFINE, 98–100, 102, 134, 152, 158–161, **242**, **287**, **390**
—.aniso_B[*m*][*n*] ($m, n = 1, 2, 3$), 158, **390**
—.B_iso_max, 158, **390**
—.B_iso_mean, 158, **391**
—.B_iso_min, 158, **391**
—.correlation_coeff_Fo_to_Fc, 158, 159, **391**
—.correlation_coeff_Fo_to_Fc_free, 158, **391**
—.details, 158, 159, *390*, **391**, 392–395
—.diff_density_max, 158, *390*, **391**
—_diff_density_max, *6*, 98, *99*, *242*, **243**, 565, 566
—.diff_density_max_esd, **391**
—.diff_density_min, 158, *390*, **391**
—_diff_density_min, *6*, 98, *99*, *242*, **243**, 565, 566
—.diff_density_min_esd, **391**
—.diff_density_rms, 158, **391**
—_diff_density_rms, 98, **243**
—.diff_density_rms_esd, **391**
—.entry_id, 158, 159, *159*, *390*, **391**
—.ls_abs_structure_details, 158, *390*, **391**
—_ls_abs_structure_details, *6*, 98, *99*, *242*, **243**, 565, 566
—.ls_abs_structure_Flack, 158, *390*, **391**
—_ls_abs_structure_Flack, 98, **243**, 565, 566
—.ls_abs_structure_Flack_esd, **391**
—.ls_abs_structure_Rogers, 158, **391**, 392
—_ls_abs_structure_Rogers, 98, **243**, 565, 566
—.ls_abs_structure_Rogers_esd, **392**
—.ls_d_res_high, 158, **392**, 393–397
—_ls_d_res_high, 98, 100, **243**, 244–247, 249
—.ls_d_res_low, 158, **392**, 393–397
—_ls_d_res_low, 98, 100, **243**, 245–247, 249
—.ls_extinction_coef, 158, *390*, **392**
—_ls_extinction_coef, *6*, 98, 99, *99*, *242*, **243**, 565
—.ls_extinction_coef_esd, **392**
—.ls_extinction_expression, 158, *390*, **392**
—_ls_extinction_expression, 98, *99*, *242*, **243**
—.ls_extinction_method, 158, *390*, **392**
—_ls_extinction_method, *6*, 98, 99, *99*, *242*, **243**, 565
—.ls_goodness_of_fit_all, 158, *390*, **392**, 395
—_ls_goodness_of_fit_all, 98, *99*, *242*, **243**
—.ls_goodness_of_fit_all_esd, **392**
—.ls_goodness_of_fit_gt, 158, **392**
—_ls_goodness_of_fit_gt, 98, *99*, *242*, **244**
—.ls_goodness_of_fit_obs, 158, *390*, **393**, 395
—_ls_goodness_of_fit_obs, *6*, 98, **244**
—.ls_goodness_of_fit_obs_esd, **393**
—.ls_goodness_of_fit_ref, 158, **393**
REFINE
—_ls_goodness_of_fit_ref, 98, **244**, 565, 566
—.ls_hydrogen_treatment, 158, *390*, **393**
—_ls_hydrogen_treatment, *6*, 98, 99, *99*, *242*, **244**, 565, 566
—.ls_matrix_type, 158, *390*, **393**
—_ls_matrix_type, 98, *99*, *242*, **244**
—_ls_mod_func_description, 134, *138*, **287**
—_ls_mod_hydrogen_treatment, 134, **287**
—_ls_mod_overall_phason_coeff, 134, **288**
—_ls_mod_overall_phason_formula, 134, **288**
—.ls_number_constraints, 158, 301, *390*, **393**
—_ls_number_constraints, 98, *99*, 212, *242*, **244**
—.ls_number_parameters, 158, *159*, *390*, **393**
—_ls_number_parameters, *6*, 98, *99*, *242*, **244**, 245, 246, 565, 566
—_ls_number_reflns, *6*, 98, *99*, *242*, **244**, 565, 566
—.ls_number_reflns_all, 158, **393**
—.ls_number_reflns_obs, 158, *159*, *390*, **393**, 395
—.ls_number_reflns_R_free, 158, **393**
—.ls_number_reflns_R_work, 158, **393**
—.ls_number_restraints, 158, *159*, 303, *390*, **394**, 395
—_ls_number_restraints, 98, *99*, 213, *242*, **244**, 245, 246
—.ls_percent_reflns_obs, 158, 159, **394**
—.ls_percent_reflns_R_free, 158, **394**
—.ls_R_factor_all, 158, *390*, **394**
—_ls_R_factor_all, 98, 99, *99*, *242*, **244**, 268
—.ls_R_factor_gt, 158, **394**, 412
—_ls_R_factor_gt, 98, 99, *99*, *242*, **245**, 251, 565, 566
—.ls_R_factor_obs, 158, 159, *159*, *390*, **394**, 395
—_ls_R_factor_obs, *6*, 98, **245**
—.ls_R_factor_R_free, 158, **394**
—.ls_R_factor_R_free_error, 158, **394**
—.ls_R_factor_R_free_error_details, 158, **394**, 406
—.ls_R_factor_R_work, 158, 159, 394, **395**
—_ls_R_Fsqd_factor, 98, *125*, **245**, 268
—.ls_R_Fsqd_factor_obs, 158, **395**
—_ls_R_I_factor, 98, **245**, 268
—.ls_R_I_factor_obs, 158, **395**
—.ls_redundancy_reflns_all, 158, **395**
—.ls_redundancy_reflns_obs, 158, **395**
—.ls_restrained_S_all, 158, 392, **395**
—_ls_restrained_S_all, 98, **245**
—_ls_restrained_S_gt, 98, **245**
—.ls_restrained_S_obs, 158, 393, **395**
—_ls_restrained_S_obs, 98, **246**
—_ls_shift/esd_max, *6*, 98, **246**
—_ls_shift/esd_mean, 98, **246**
—_ls_shift/su_max, 98, *99*, *242*, **246**, 565, 566
—_ls_shift/su_max_lt, 98, **246**
—_ls_shift/su_mean, 98, *99*, *242*, **246**
—_ls_shift/su_mean_lt, 98, **246**
—.ls_shift_over_esd_max, 158, *390*, **395**
—.ls_shift_over_esd_mean, 158, *390*, **396**
—.ls_shift_over_su_max, 158, **396**
—.ls_shift_over_su_max_lt, 158, **396**
—.ls_shift_over_su_mean, 158, **396**
—.ls_shift_over_su_mean_lt, 158, **396**
—.ls_structure_factor_coef, 158, *390*, 392, 393, 395, **396**, 397, 403, 406, 407, 413
—_ls_structure_factor_coef, *6*, 98, 99, *99*, *242*, 243–245, **246**, 247, 248, 252, 565, 566
—.ls_weighting_details, 158, *159*, *390*, 396, **396**
—_ls_weighting_details, 98, 99, *99*, *242*, **246**, 565
—.ls_weighting_scheme, 158, *159*, *390*, **396**
—_ls_weighting_scheme, *6*, 99, *99*, *242*, **246**, 565, 567
—.ls_wR_factor_all, 158, *390*, **396**
—_ls_wR_factor_all, 99, *99*, *242*, **246**
—_ls_wR_factor_gt, 99, *99*, *242*, **246**
—.ls_wR_factor_obs, 158, *390*, **396**
—_ls_wR_factor_obs, *6*, 99, **247**
REFINE
—.ls_wR_factor_R_free, 158, **396**
—.ls_wR_factor_R_work, 158, **397**
—_ls_wR_factor_ref, 99, **247**, 565, 566
—.occupancy_max, 158, **397**
—.occupancy_min, 158, **397**
—.overall_FOM_free_R_set, 158, **397**
—.overall_FOM_work_R_set, 158, **397**
—.overall_SU_B, 158, 159, **397**
—.overall_SU_ML, 158, 159, **397**
—.overall_SU_R_Cruickshank_DPI, 158, 159, **398**
—.overall_SU_R_free, 158, 159, **398**
—.solvent_model_details, 158, **398**
—.solvent_model_param_bsol, 158, **398**
—.solvent_model_param_ksol, 158, **398**
—_special_details, 99, *99*, *242*, 243, 244, **247**
REFINE_ANALYZE, 158–160, **398**
—.entry_id, 159, *160*, **398**, *398*
—.Luzzati_coordinate_error_free, 160, **398**
—.Luzzati_coordinate_error_obs, 160, *160*, **398**, *398*
—.Luzzati_d_res_low_free, 160, **398**
—.Luzzati_d_res_low_obs, 160, *160*, **398**, *398*
—.Luzzati_sigma_a_free, 160, **399**
—.Luzzati_sigma_a_free_details, 160, **399**
—.Luzzati_sigma_a_obs, 160, **399**
—.Luzzati_sigma_a_obs_details, 160, **399**
—.number_disordered_residues, 160, **399**
—.occupancy_sum_hydrogen, 160, **399**
—.occupancy_sum_non_hydrogen, 160, **399**
—.RG_d_res_high, 160, **399**
—.RG_d_res_low, 160, **399**, 400
—.RG_free, 160, **399**
—.RG_free_work_ratio, 160, **399**
—.RG_work, 160, 399, **400**
REFINE_B_ISO, 158, 161, 162, **400**
—.class, 161, *161*, 162, **400**, *400*
—.details, 161, **400**
—.treatment, 161, *161*, **400**, *400*
—.value, 161, **400**
REFINE_FUNCT_MINIMIZED, 158, 159, **400**
—.number_terms, 158, *159*, **400**, *400*
—.residual, 158, *159*, **400**, *400*
—.type, 158, *159*, **400**, *400*
—.weight, 158, **400**
REFINE_HIST, 158, 159, 162, **400**, 401
—.cycle_id, 162, *162*, **401**, *401*
—.d_res_high, 162, *162*, **401**, *401*, 402
—.d_res_low, 162, *162*, **401**, *401*
—.details, 162, *162*, **401**, *401*
—.number_atoms_solvent, 162, *162*, **401**, *401*
—.number_atoms_total, 162, *162*, **401**, *401*
—.number_reflns_all, 162, *162*, **401**, *401*
—.number_reflns_obs, 162, *162*, **401**, *401*
—.number_reflns_R_free, 162, *162*, **401**, *401*
—.number_reflns_R_work, 162, *162*, **401**, *401*
—.R_factor_all, 162, *162*, **401**, *401*
—.R_factor_obs, 162, *162*, **401**, *401*
—.R_factor_R_free, 162, *162*, **401**, *401*
—.R_factor_R_work, 162, *162*, *401*, **402**
REFINE_LS_CLASS, 98–100, 158, 160, 161, 164, **247**, **402**
—.code, 160, **402**
—_code, 99, 100, **247**, *247*
—.d_res_high, 160, **402**, 403
—_d_res_high, 99, **247**
—.d_res_low, 160, **402**, 403
—_d_res_low, 99, **247**, 248
—.R_factor_all, 160, **402**
—_R_factor_all, 99, **247**
—.R_factor_gt, 160, **402**
—_R_factor_gt, 99, **247**, *247*
—.R_Fsqd_factor, 160, **402**
—_R_Fsqd_factor, 99, **247**
—.R_I_factor, 160, **402**
—_R_I_factor, 99, **248**
—.wR_factor_all, 160, 402, **403**
—_wR_factor_all, 99, **248**

REFINE_LS_RESTR, 158, 160, 161, **403**, 404
—.criterion, 160, *161*, **403**, *403*
—.dev_ideal, 160, *161*, **403**, *403, 404*
—.dev_ideal_target, 160, *161*, **403**, *403, 404*
—.number, 160, *161*, **403**, *403, 404*
—.rejects, 160, *161*, **403**, *403*
—.type, 160, 161, *161*, **403**, *403*, 404, *404*
—.weight, 160, **403**
REFINE_LS_RESTR_NCS, 158, 160, 161, **404**
—.dom_id, 160, **404**, *404*
—.ncs_model_details, 160, 161, **404**, *404*
—.rms_dev_B_iso, 160, **404**, *404*
—.rms_dev_position, 160, **404**, *404*
—.weight_B_iso, 160, **404**, *404*
—.weight_position, 160, **404**, *404*
REFINE_LS_RESTR_TYPE, 158, 160, 161, **404**
—.distance_cutoff_high, 160, **404**, *404*
—.distance_cutoff_low, 160, **404**, *404*
—.type, 160, **404**, *404*
REFINE_LS_SHELL, 158, 160, 161, **405**
—.d_res_high, 160, *161*, **405**, *405*, 406, 407
—.d_res_low, 160, *161*, **405**, *405*, 406, 407
—.number_reflns_all, 160, **405**
—.number_reflns_obs, 160, *161*, **405**, *405*
—.number_reflns_R_free, 160, **405**
—.number_reflns_R_work, 160, **405**
—.percent_reflns_obs, 160, **405**
—.percent_reflns_R_free, 160, **405**
—.R_factor_all, 160, **405**
—.R_factor_obs, 160, *161*, **405**, *405*
—.R_factor_R_free, 160, **406**
—.R_factor_R_free_error, 160, **406**
—.R_factor_R_work, 160, **406**
—.redundancy_reflns_all, 160, **406**
—.redundancy_reflns_obs, 160, **406**
—.wR_factor_all, 160, **406**
—.wR_factor_obs, 160, **406**
—.wR_factor_R_free, 160, **406**
—.wR_factor_R_work, 160, **407**
REFINE_OCCUPANCY, 158, 161, 162, **407**
—.class, 161, *161*, 162, **407**, *407*
—.details, 161, *161*, **407**, *407*
—.treatment, 161, *161*, **407**, *407*
—.value, 161, *161*, **407**, *407*
REFLN, 96–98, 100, 101, 118, 121, 122, 134, 152, 162, 163, **248**, 249, 250, 252, **288**, 347, 349, 350, **407**, 409–411, 413
_pd_refln_peak_id, 122, 123, 129, 130, 265, **268**
_pd_refln_phase_id, 122, 123, *125*, 129, 130, 265, 266, **268**
_pd_refln_wavelength_id, 122, 123, 265, **268**
—.A_calc, 162, **407**
—._A_calc, 100, **248**
—.A_calc_au, 162, **407**
—.A_meas, 162, **407**
—._A_meas, 100, **248**
—.A_meas_au, 162, **407**
—.B_calc, 162, **408**
—._B_calc, 100, **248**
—.B_calc_au, 162, **408**
—.B_meas, 162, **408**
—._B_meas, 100, **248**
—.B_meas_au, 162, **408**
—.class_code, 162, **408**
—._class_code, 100–102, **248**
—.crystal_id, 162, **408**
—._crystal_id, 100, 235, **248**
—.d_spacing, 162, **408**
—._d_spacing, 100, 123, *125*, **248**
—.F_calc, 162, **408**
—._F_calc, 100, *248*, **249**, 566
—.F_calc_au, 162, **408**
—.F_meas, 162, **408**
—._F_meas, 100, *248*, **249**, 566
—.F_meas_au, 162, **408**
—.F_meas_sigma, 162, **408**
—.F_meas_sigma_au, 162, **408**
—._F_sigma, 100, *248*, **249**, 566
—.F_squared_calc, 162, *163*, *407*, **408**
REFLN
—._F_squared_calc, 100, *100*, 123, *125*, 129, **249**, 566
—.F_squared_meas, 162, *163*, *407*, **408**
—._F_squared_meas, 100, *100*, 123, *125*, **249**, 566
—.F_squared_sigma, 162, *163*, *407*, **408**
—._F_squared_sigma, 100, *100*, **249**, 566
—.fom, 162, 163, **408**
—.include_status, 162, **408**
—._include_status, 100, *100*, 101, *248*, **249**
—.index_h, 162, *163*, *407*, **409**
—._index_h, 77, 100, *100*, 123, *125*, 129, 217, 229, 230, *248*, **249**, 566
—.index_k, 162, *163*, *407*, **409**
—._index_k, 77, 100, *100*, 123, *125*, 129, *248*, **249**, 566
—.index_l, 162, *163*, *407*, **409**
—._index_l, 77, 100, *100*, 123, *125*, 129, *248*, **249**, 566
—._index_m_*n* ($n = 1, \ldots, 8$), 134, **288**
—.intensity_calc, 162, **409**
—._intensity_calc, 100, 123, **249**
—.intensity_meas, 162, **409**
—._intensity_meas, 100, 123, **249**
—.intensity_sigma, 162, **409**
—._intensity_sigma, 100, **249**
—.mean_path_length_tbar, 162, **409**
—._mean_path_length_tbar, 100, **249**
—._observed_status, 100, *125*, **249**
—.phase_calc, 162, **409**
—._phase_calc, 100, *125*, **249**
—.phase_meas, 162, **409**
—._phase_meas, 100, **249**
—.refinement_status, 162, **409**
—._refinement_status, 100, **249**
—.scale_group_code, 162, **409**, 413
—._scale_group_code, 100–102, *248*, **249**, 252
—._sint/lambda, 100, 248, **249**
—.sint_over_lambda, 162, **409**
—.status, 162, *163*, *407*, **409**
—.symmetry_epsilon, 162, **409**
—._symmetry_epsilon, 100, **249**
—.symmetry_multiplicity, 162, **409**
—._symmetry_multiplicity, 100, **249**
—.wavelength, 162, **409**
—._wavelength, 100, **250**
—.wavelength_id, 163, **409**
—._wavelength_id, 100, 101, 123, **250**
REFLN_SYS_ABS, 162, 163, **410**
—.I, 163, **410**, *410*
—.I_over_sigmaI, 163, **410**, *410*
—.index_h, 163, **410**, *410*
—.index_k, 163, **410**, *410*
—.index_l, 163, **410**, *410*
—.sigmaI, 163, **410**, *410*
REFLNS, 98, 100–102, 134, 162–164, 248, **250**, **288**, 407, **410**
—.B_iso_Wilson_estimate, 163, **410**
—.d_resolution_high, 163, *164*, **410**, *410*, 412
—._d_resolution_high, 101, **250**, *250*
—.d_resolution_low, 163, *164*, **410**, *410*, 412
—._d_resolution_low, 101, **250**, *250*
—.data_reduction_details, 163, *164*, **410**, *410*
—.data_reduction_method, 163, *164*, *410*, **411**
—.details, 163, *164*, *410*, **411**, 413
—.entry_id, 163, *164*, *410*, **411**
—.Friedel_coverage, 163, **411**
—._Friedel_coverage, 101, **250**
—.limit_h_max, 163, *164*, *410*, **411**
—._limit_h_max, 101, **250**, *250*
—.limit_h_min, 163, *164*, *410*, **411**
—._limit_h_min, 101, **250**, *250*
—._limit_index_m_*n*_max ($n = 1, \ldots, 8$), 134, **288**
—._limit_index_m_*n*_min ($n = 1, \ldots, 8$), 134, **288**
—.limit_k_max, 163, *164*, *410*, **411**
—._limit_k_max, 101, **250**, *250*
REFLNS
—.limit_k_min, 163, *164*, *410*, **411**
—._limit_k_min, 101, **250**, *250*
—.limit_l_max, 163, *164*, *410*, **411**
—._limit_l_max, 101, **250**, *250*
—.limit_l_min, 163, *164*, *410*, **411**
—._limit_l_min, 101, **250**, *250*
—.number_all, 163, *410*, **411**
—.number_gt, 163, 394, **411**, 412
—._number_gt, 101, 245, **250**, *250*, 251, 565, 566
—.number_obs, 163, *164*, *410*, **411**
—._number_observed, *6*, 101, *125*, **250**
—._number_total, *6*, 101, **250**, *250*, 565–567
—.observed_criterion, 163, 164, 393–397, 401, 402, 405–407, *410*, **411**, 412, 414, 415
—._observed_criterion, *6*, 100, 101, 244, 245, 247, **250**, 252–254
—.observed_criterion_F_max, 163, **411**
—.observed_criterion_F_min, 163, **412**
—.observed_criterion_I_max, 163, **412**
—.observed_criterion_I_min, 163, **412**
—.observed_criterion_sigma_F, 163, **412**
—.observed_criterion_sigma_I, 163, *164*, **412**
—.percent_possible_obs, 163, **412**
—.R_free_details, 159, 163, 394–397, 401, 402, 405–407, **412**
—.Rmerge_F_all, 163, **412**
—.Rmerge_F_obs, 163, **412**
—._special_details, 101, 250, **251**
—.threshold_expression, 163, 394, 402, 411, **412**, 413–415
—._threshold_expression, 99, 101, 244–250, *250*, **251**, 252–254, 565, 566
REFLNS_CLASS, 98, 100–102, 162–164, **251**, **412**
—.code, 163, 402, 408, **412**
—._code, 100–102, 247, 248, **251**, *251*
—.d_res_high, 163, **412**, 413
—._d_res_high, 101, **251**, 252
—.d_res_low, 163, **412**, 413
—._d_res_low, 101, **251**, 252
—.description, 163, **412**
—._description, 101, 102, **251**
—.number_gt, 163, **413**
—._number_gt, 101, 102, **251**, *251*
—.number_total, 163, **413**
—._number_total, 101, **251**
—.R_factor_all, 163, **413**
—._R_factor_all, 101, **251**
—.R_factor_gt, 163, **413**
—._R_factor_gt, 101, **251**
—.R_Fsqd_factor, 163, **413**
—._R_Fsqd_factor, 101, **251**
—.R_I_factor, 163, **413**
—._R_I_factor, 101, **252**
—.wR_factor_all, 163, **413**
—._wR_factor_all, 99, 101, **252**
REFLNS_SCALE, 98, 100–102, 162–164, **252**, 409, **413**
—.group_code, 163, 409, **413**, *413*, 414
—._group_code, 101, 102, 249, **252**, *252*
—.meas_F, 163, 413, *413*, **414**
—._meas_F, 101, **252**, *252*
—.meas_F_squared, 163, **414**
—._meas_F_squared, 101, **252**
—.meas_intensity, 163, 413, **414**
—._meas_intensity, 101, **252**
REFLNS_SHELL, 98, 100–102, 162–164, **252**, **414**
—.d_res_high, 163, *164*, 384, **414**, *414*, 415
—._d_res_high, 101, 102, **252**, *252*
—.d_res_low, 163, *164*, 384, **414**, *414*, 415
—._d_res_low, 101, **252**, *252*
—.meanI_over_sigI_all, 163, **414**
—._meanI_over_sigI_all, 101, **252**
—.meanI_over_sigI_gt, 163, **414**
—._meanI_over_sigI_gt, 101, **252**
—.meanI_over_sigI_obs, 163, *164*, **414**, *414*
—._meanI_over_sigI_obs, 101, **252**

REFLNS_SHELL
—.meanI_over_uI_all, 163, **414**
—._meanI_over_uI_all, 101, **253**
—.meanI_over_uI_gt, 163, **414**
—._meanI_over_uI_gt, 101, *252*, **253**
—.number_measured_all, 163, **414**
—._number_measured_all, 101, **253**
—.number_measured_gt, 163, **414**
—._number_measured_gt, 101, *252*, **253**
—.number_measured_obs, 163, *164*, **414**, *414*
—._number_measured_obs, 101, **253**
—.number_possible, 163, **414**
—._number_possible, 101, **253**
—.number_unique_all, 163, **414**
—._number_unique_all, 101, **253**
—.number_unique_gt, 163, **414**
—._number_unique_gt, 101, *252*, **253**
—.number_unique_obs, 163, *164*, *414*, **415**
—._number_unique_obs, 101, **253**
—.percent_possible_all, 163, **415**
—._percent_possible_all, 101, **253**
—.percent_possible_gt, 163, **415**
—._percent_possible_gt, 101, *252*, **253**
—.percent_possible_obs, 163, *164*, *414*, **415**
—._percent_possible_obs, 101, **253**
—.Rmerge_F_all, 163, **415**
—._Rmerge_F_all, 101, **253**
—.Rmerge_F_gt, 163, **415**
—._Rmerge_F_gt, 101, *252*, **253**
—.Rmerge_F_obs, 163, *164*, *414*, **415**
—._Rmerge_F_obs, 101, **254**
—.Rmerge_I_all, 163, **415**
—._Rmerge_I_all, 101, **254**
—.Rmerge_I_gt, 163, **415**
—._Rmerge_I_gt, 101, **254**
—.Rmerge_I_obs, 163, **415**
—._Rmerge_I_obs, 101, **254**
_related_function, *see* DDL1
_related_item, *see* DDL1

SOFTWARE, 91, 113, 153, 154, 190, 191, 337, 374, 375, 378, 411, **416**
—.citation_id, 191, *191*, **416**, *416*
—.classification, 191, *191*, **416**, *416*
—.compiler_name, 191, *191*, **416**, *416*
—.compiler_version, 191, *191*, **416**, *416*
—.contact_author, 191, *191*, **416**, *416*
—.contact_author_email, 191, *191*, **416**, *416*
—.date, 191, *191*, **416**, *416*
—.dependencies, 191, *191*, **416**, *416*
—.description, 191, *191*, **416**, *416*
—.hardware, 191, *191*, **416**, *416*
—.language, 191, *191*, **416**, *416*
—.location, 191, *191*, **416**, *416*
—.mods, 191, *191*, **416**, *416*
—.name, 191, *191*, **416**, *416*
—.os, 191, *191*, **416**, *416*
—.os_version, 191, *191*, **416**, *416*
—.type, 191, *191*, *416*, **417**
—.version, 191, *191*, *416*, **417**
SPACE_GROUP, 110, 111, 132, 134, 136, 190, 207, **254**, **288**, **417**, **459**
—.Bravais_type, 207, **459**, *459*
—.centring_type, 207, *459*, **460**
—.crystal_system, 190, 207, **417**, *417*, *459*, **460**
—._crystal_system, 111, **254**, *254*
—.id, 190, 207, **417**, *417*, 418, *459*, **460**, 466
—._id, 111, **254**, *254*, 255
—.IT_coordinate_system_code, **459**
—.IT_number, 190, 207, **417**, *417*, *459*, **460**, 461
—._IT_number, 111, **254**, *254*, 255
—.Laue_class, 207, *459*, **460**
—.name_H-M_alt, 190, 206, 207, **417**, *417*, **460**, 461
—._name_H-M_alt, 110, 111, **254**, *254*
—.name_H-M_alt_description, 207, **460**
—.name_H-M_full, 207, **460**
—.name_H-M_ref, 206, 207, *459*, **461**
SPACE_GROUP
—.name_Hall, 190, 207, **417**, *417*, *459*, 460, 461, **462**
—._name_Hall, 111, *254*, **255**
—.name_Schoenflies, 207, *459*, 460, 461, **462**
—.Patterson_name_H-M, 207, *459*, **460**
—.point_group_H-M, 207, **462**
—.reference_setting, 207, 460, 461, **462**, 465
—._ssg_IT_number, 136, **288**
—._ssg_name, 136, **288**
—._ssg_name_IT, 136, *138*, **288**
—._ssg_name_WJJ, 136, *138*, 288, **289**
—._ssg_WJJ_code, 136, **289**
—.transform_Pp_abc, 207, 462, **465**
—.transform_Qq_xyz, 207, 462, **465**
SPACE_GROUP_SYMOP, 110, 111, 132, 134, 136, 190, 207, **255**, **289**, **418**, **465**, 466
—.generator_xyz, 207, **465**
—.id, 190, 207, **418**, *418*, **465**, *465*
—._id, 111, *111*, 236–238, **255**, *255*
—.operation_description, 207, **465**, *465*
—.operation_xyz, 190, 207, **418**, *418*, **465**, *465*, 466
—._operation_xyz, 111, *111*, **255**, *255*
—.sg_id, 190, 207, **418**, **466**
—._sg_id, 111, **255**
—._ssg_id, 136, *138*, 286, 287, **289**, *289*
—._ssg_operation_algebraic, 136, *138*, **289**, *289*
SPACE_GROUP_WYCKOFF, 207, **466**
—.coords_xyz, 207, **466**
—.id, 207, **466**, *466*
—.letter, 207, **466**, *466*
—.multiplicity, 207, **466**, *466*
—.sg_id, 207, 208, **466**
—.site_symmetry, 207, **466**, *466*
_stereo_atom_id, *see* MIF
_stereo_bond_id_1, *see* MIF
_stereo_bond_id_2, *see* MIF
_stereo_geometry, *see* MIF
_stereo_vertex_id, *see* MIF
STRUCT, 91, 145, 164, 166, 174–176, 178–180, **418**
—.entry_id, *5*, 179, **418**, *418*
—.title, *5*, 179, 180, **418**, *418*
STRUCT_ASYM, 145, 166, 174, 178–180, 302, 352, **418**, 419, 438
—.details, *145*, 179, *180*, **418**, *418*
—.entity_id, *145*, 179, *180*, **418**, *418*
—.id, *145*, 166, 179, 180, *180*, 181, 302, **418**, *418*, 419, 438, *439*
STRUCT_BIOL, 145, 178–180, 353, **419**, 420, 433
—.details, *146*, 179, 180, *180*, **419**, *419*
—.id, *146*, 179, *180*, **419**, *419*, 420, 433
STRUCT_BIOL_GEN, 178–181, **419**, 420, 421
—.asym_id, *146*, 179, *180*, **419**, *419*
—.biol_id, *146*, 179, *180*, **419**, *419*
—.details, 179, **419**
—.symmetry, *146*, 179, *180*, **419**, *419*
STRUCT_BIOL_KEYWORDS, 178–180, **419**
—.biol_id, 179, **420**, *420*
—.text, 179, **420**, *420*
STRUCT_BIOL_VIEW, 178, 179, 181, **420**
—.biol_id, 179, **420**, *420*
—.details, 179, 181, **420**, *420*, 421
—.id, 179, **420**, *420*
—.rot_matrix[m][n] ($m, n = 1, 2, 3$), 179, **420**, *420*, **421**
STRUCT_CONF, 145, 178, 181, **421**, 422
—.beg_auth_asym_id, 181, **421**
—.beg_auth_comp_id, 181, **421**
—.beg_auth_seq_id, 181, **421**
—.beg_label_asym_id, *146*, 181, *181*, **421**, *421*
—.beg_label_comp_id, *146*, 181, *181*, **421**, *421*
—.beg_label_seq_id, *146*, 181, *181*, **421**, *421*
—.conf_type_id, *146*, 181, *181*, **421**, *421*
—.details, *146*, 181, 182, **421**, *421*
—.end_auth_asym_id, 181, **422**
—.end_auth_comp_id, 181, **422**
—.end_auth_seq_id, 181, **422**
STRUCT_CONF
—.end_label_asym_id, *146*, 181, *181*, *421*, **422**
—.end_label_comp_id, *146*, 181, *181*, *421*, **422**
—.end_label_seq_id, *146*, 181, *181*, *421*, **422**
—.id, *146*, 181, *181*, *421*, **422**
STRUCT_CONF_TYPE, 178, 181, 421, **422**
—.criteria, 181, *181*, 182, **422**, *422*
—.id, 181, *181*, 182, 421, **422**, *422*
—.reference, 181, 182, **422**, *422*
STRUCT_CONN, 146, 173, 178, 182, 183, **423**
—.conn_type_id, *146*, 182, *183*, **423**, *423*
—.details, *146*, 182, **423**, *423*
—.id, *146*, 182, *183*, **423**, *423*
—.ptnr1_auth_asym_id, 182, **423**
—.ptnr1_auth_atom_id, 182, **423**
—.ptnr1_auth_comp_id, 182, **423**
—.ptnr1_auth_seq_id, 182, **423**
—.ptnr1_label_alt_id, 182, **423**
—.ptnr1_label_asym_id, *146*, 182, *183*, **423**, *423*
—.ptnr1_label_atom_id, *146*, 182, *183*, **423**, *423*
—.ptnr1_label_comp_id, *146*, 182, *183*, **423**, *423*
—.ptnr1_label_seq_id, *146*, 182, *183*, **423**, *423*
—.ptnr1_role, *146*, 182, *183*, **423**, *423*
—.ptnr1_symmetry, *146*, 182, *183*, **423**, *423*
—.ptnr2_auth_asym_id, 182, **423**
—.ptnr2_auth_atom_id, 182, **423**
—.ptnr2_auth_comp_id, 182, **423**
—.ptnr2_auth_seq_id, 182, **423**
—.ptnr2_label_alt_id, 182, **424**
—.ptnr2_label_asym_id, *146*, 182, *183*, *423*, **424**
—.ptnr2_label_atom_id, *146*, 182, *183*, *423*, **424**
—.ptnr2_label_comp_id, *146*, 182, *183*, *423*, **424**
—.ptnr2_label_seq_id, *146*, 182, *183*, *423*, **424**
—.ptnr2_role, *146*, 182, *183*, *423*, **424**
—.ptnr2_symmetry, *146*, 182, *183*, *423*, **424**
STRUCT_CONN_TYPE, 178, 182, 183, 423, **424**
—.criteria, 182, 183, *183*, **424**, *424*
—.id, 182, *183*, 423, **424**, *424*
—.reference, 182, 183, **424**, *424*
STRUCT_KEYWORDS, 179, 180, **424**
—.entry_id, 179, **424**, *424*
—.text, 179, **424**, *424*
STRUCT_MON_DETAILS, 178, 183, 184, **424**, 425, 428
—.entry_id, 183, **424**
—.prot_cis, 183, 184, **424**, 430
—.RSCC, 183, **425**, 426, 427, 429
—.RSR, 183, 184, **425**, 427, 429, 430
STRUCT_MON_NUCL, 178, 183, 184, 424, **425**, 428
—.alpha, 183, **425**, *425*
—.auth_asym_id, 183, **425**
—.auth_comp_id, 183, **425**
—.auth_seq_id, 183, **425**
—.beta, 183, **425**, *425*
—.chi$n$ ($n = 1, 2$), 183, **425**
—.delta, 183, *425*, **426**
—.details, 183, **426**
—.epsilon, 183, *425*, **426**
—.gamma, 183, *425*, **426**
—.label_alt_id, 183, *425*, **426**
—.label_asym_id, 183, *425*, **426**
—.label_comp_id, 183, *425*, **426**
—.label_seq_id, 183, *425*, **426**
—.mean_B_all, 183, **426**
—.mean_B_base, 183, **426**
—.mean_B_phos, 183, **426**
—.mean_B_sugar, 183, **426**
—.nu$n$ ($n = 0, \ldots, 4$), 183, **426**
—.P, 183, **426**
—.RSCC_all, 183, **426**
—.RSCC_base, 183, **427**
—.RSCC_phos, 183, **427**
—.RSCC_sugar, 183, **427**
—.RSR_all, 183, **427**
—.RSR_base, 183, **427**
—.RSR_phos, 183, **427**
—.RSR_sugar, 183, **427**
—.tau$n$ ($n = 0, \ldots, 4$), 183, **428**

STRUCT_MON_NUCL
—.taum, 183, **428**
—.zeta, 183, *425*, **428**
STRUCT_MON_PROT, 178, 183, 184, 424, 425, **428**
—.auth_asym_id, 183, **428**
—.auth_comp_id, 183, **428**
—.auth_seq_id, 183, **428**
—.chi*n* ($n = 1, \ldots, 5$), 183, *184*, **428**, *428*
—.details, 183, **429**
—.label_alt_id, 183, *184*, *428*, **429**
—.label_asym_id, 183, *184*, *428*, **429**
—.label_comp_id, 183, *184*, *428*, **429**
—.label_seq_id, 183, *184*, *428*, **429**
—.mean_B_all, 183, *184*, *428*, **429**
—.mean_B_main, 183, *184*, *428*, **429**
—.mean_B_side, 183, *184*, *428*, **429**
—.omega, 183, *184*, *428*, **429**
—.phi, 183, *184*, *428*, **429**
—.psi, 183, *184*, *428*, **429**
—.RSCC_all, 183, *184*, 425, *428*, **429**
—.RSCC_main, 183, 425, **429**
—.RSCC_side, 183, **429**
—.RSR_all, 183, *184*, 425, *428*, **429**
—.RSR_main, 183, 425, **430**
—.RSR_side, 183, 425, **430**
STRUCT_MON_PROT_CIS, 178, 183, 184, **430**
—.auth_asym_id, 183, **430**
—.auth_comp_id, 183, **430**
—.auth_seq_id, 183, **430**
—.label_alt_id, 183, **430**, *430*
—.label_asym_id, 183, **430**, *430*
—.label_comp_id, 183, **430**, *430*
—.label_seq_id, 183, **430**, *430*
STRUCT_NCS_DOM, 161, 178, 184–186, 404, **430**, 431, 432
—.details, 184, **430**, *430*
—.id, 184, *186*, 404, *430*, **431**, 432
STRUCT_NCS_DOM_LIM, 178, 184–186, **431**
—.beg_auth_asym_id, 185, **431**
—.beg_auth_comp_id, 185, **431**
—.beg_auth_seq_id, 185, **431**
—.beg_label_alt_id, 184, *186*, **431**, *431*
—.beg_label_asym_id, 184, *186*, **431**, *431*
—.beg_label_comp_id, 184, *186*, **431**, *431*
—.beg_label_seq_id, 185, *186*, **431**, *431*
—.dom_id, 185, *186*, **431**, *431*
—.end_auth_asym_id, 185, **431**
—.end_auth_comp_id, 185, **431**
—.end_auth_seq_id, 185, **431**
—.end_label_alt_id, 185, *186*, **431**, *431*
—.end_label_asym_id, 185, *186*, **431**, *431*
—.end_label_comp_id, 185, *186*, **431**, *431*
—.end_label_seq_id, 185, *186*, **431**, *431*
STRUCT_NCS_ENS, 178, 184–186, **431**, 432
—.details, 184, *431*, **432**
—.id, 184, 185, *186*, *431*, **432**
—.point_group, 184, *186*, **432**
STRUCT_NCS_ENS_GEN, 179, 184–186, **432**
—.dom_id_1, 184, *186*, **432**, *432*
—.dom_id_2, 184, *186*, **432**, *432*
—.ens_id, 184, *186*, **432**, *432*
—.oper_id, 184, *186*, **432**, *432*
STRUCT_NCS_OPER, 179, 185, 186, **432**
—.code, 185, 186, **432**, *432*
—.details, 185, **432**, *432*
—.id, 185, **432**, *432*
—.matrix[m][n] ($m, n = 1, 2, 3$), 185, **432**, *432*, **433**
—.vector[1], 185, *432*, **433**
—.vector[2], 185, *432*, **433**
—.vector[3], 185, *432*, **433**
STRUCT_REF, 179, 186, 187, 192, **433**, 434
—.biol_id, 186, *186*, 187, **433**, *433*
—.db_code, 186, *186*, **433**, *433*
—.db_name, 186, *186*, **433**, *433*
—.details, 186, **433**, *433*
—.entity_id, 186, *186*, 187, **433**, *433*
—.id, 186, *186*, 187, **433**, *433*, 434
STRUCT_REF
—.seq_align, 186, *186*, **433**, *433*
—.seq_dif, 186, *186*, 433, *433*, **434**
STRUCT_REF_SEQ, 179, 186, 433, **434**
—.align_id, 186, *186*, **434**, *434*
—.db_align_beg, 186, *186*, **434**, *434*
—.db_align_end, 186, *186*, **434**, *434*
—.details, 186, *186*, **434**, *434*
—.ref_id, 186, *186*, **434**, *434*
—.seq_align_beg, 186, *186*, **434**, *434*
—.seq_align_end, 186, *186*, **434**, *434*
STRUCT_REF_SEQ_DIF, 179, 186, 433, **434**
—.align_id, 186, **434**, *434*
—.db_mon_id, 186, **434**, *434*
—.details, 186, **434**, *434*
—.mon_id, 186, **434**, *434*
—.seq_num, 186, **434**, *434*
STRUCT_SHEET, 145, 179, 187, 188, **435**, 436–439
—.details, 187, *435*, **436**
—.id, 187, *188*, *435*, **436**, 437–439
—.number_strands, 187, *188*, *435*, **436**
—.type, 187, *435*, **436**
STRUCT_SHEET_HBOND, 179, 187, 188, **436**
—.range_1_beg_auth_atom_id, 187, **436**
—.range_1_beg_auth_seq_id, 187, **436**
—.range_1_beg_label_atom_id, 187, *188*, **436**, *436*
—.range_1_beg_label_seq_id, 187, *188*, **436**, *436*
—.range_1_end_auth_atom_id, 187, **436**
—.range_1_end_auth_seq_id, 187, **436**
—.range_1_end_label_atom_id, 187, **436**, *436*
—.range_1_end_label_seq_id, 187, **436**, *436*
—.range_2_beg_auth_atom_id, 187, **437**
—.range_2_beg_auth_seq_id, 187, **437**
—.range_2_beg_label_atom_id, 187, *188*, *436*, **437**
—.range_2_beg_label_seq_id, 187, *188*, *436*, **437**
—.range_2_end_auth_atom_id, 187, **437**
—.range_2_end_auth_seq_id, 187, **437**
—.range_2_end_label_atom_id, 187, *436*, **437**
—.range_2_end_label_seq_id, 187, *436*, **437**
—.range_id_1, 187, *188*, *436*, **437**
—.range_id_2, 187, *188*, *436*, **437**
—.sheet_id, 187, *188*, *436*, **437**
STRUCT_SHEET_ORDER, 179, 187, 188, **437**, 439
—.offset, 187, **437**, *437*
—.range_id_1, 187, *188*, **437**, *437*
—.range_id_2, 187, *188*, **437**, *437*
—.sense, 187, *188*, **437**, *437*
—.sheet_id, 187, *188*, **437**, *437*
STRUCT_SHEET_RANGE, 179, 187, 188, 437, **438**, 439
—.beg_auth_asym_id, 187, **438**
—.beg_auth_comp_id, 187, **438**
—.beg_auth_seq_id, 187, **438**
—.beg_label_asym_id, 187, *188*, **438**, *438*
—.beg_label_comp_id, 187, *188*, **438**, *438*
—.beg_label_seq_id, 187, *188*, **438**, *438*
—.end_auth_asym_id, 187, **438**
—.end_auth_comp_id, 187, **438**
—.end_auth_seq_id, 187, **438**
—.end_label_asym_id, 187, *188*, **438**, *438*
—.end_label_comp_id, 187, *188*, **438**, *438*
—.end_label_seq_id, 187, *188*, **438**, *438*
—.id, 187, *188*, 437, **438**, *438*, 439
—.sheet_id, 187, *188*, **438**, *438*
—.symmetry, 187, 188, **438**, *438*
STRUCT_SHEET_TOPOLOGY, 179, 187, 437, **439**
—.offset, 187, **439**, *439*
—.range_id_1, 187, **439**, *439*
—.range_id_2, 187, **439**, *439*
—.sense, 187, **439**, *439*
—.sheet_id, 187, **439**, *439*
STRUCT_SITE, 82, 188, 189, **439**, 440, 441
—.details, 82, 188, 189, *189*, **439**, *439*
—.id, 82, 188, 189, *189*, **439**, *439*, 440, 441
STRUCT_SITE_GEN, 82, 179, 189, **439**, 440, 441
—.auth_asym_id, 189, **439**
—.auth_atom_id, 189, **440**
—.auth_comp_id, 189, **440**
—.auth_seq_id, 189, **440**
—.details, 82, 189, *439*, **440**
—.id, 82, 189, *189*, *439*, **440**
—.label_alt_id, 189, **440**
—.label_asym_id, 82, 189, *189*, *439*, **440**
—.label_atom_id, 189, **440**
—.label_comp_id, 82, 189, *189*, *439*, **440**
—.label_seq_id, 82, 189, *189*, *439*, **440**
—.site_id, 82, 189, *189*, *439*, **440**
—.symmetry, 82, 189, *189*, *439*, **440**
STRUCT_SITE_KEYWORDS, 179, 180, 189, **440**
—.site_id, 189, *189*, **440**, *440*
—.text, 189, *189*, **440**, *440*
STRUCT_SITE_VIEW, 179, 189, **440**
—.details, 189, *189*, **440**, *440*, 441
—.id, 189, *189*, **440**, *440*
—.rot_matrix[m][n] ($m, n = 1, 2, 3$), 189, *189*, *440*, **441**
—.site_id, 189, *189*, **441**
_sub_category.description, *see* DDL2
_sub_category.id, *see* DDL2
_sub_category_examples.case, *see* DDL2
_sub_category_examples.detail, *see* DDL2
_sub_category_examples.id, *see* DDL2
_sub_category_methods.method_id, *see* DDL2
_sub_category_methods.sub_category_id, *see* DDL2
SUB_CATEGORY, 70, **477**
SUB_CATEGORY_EXAMPLES, 70, **478**
SUB_CATEGORY_METHODS, 70, **478**
SYMMETRY, 110, 134, 164, 190, 207, 236–238, **255**, 286–288, 362, 364, 366, 367, 369, **441**
—.cell_setting, 190, *441*, **442**
—._cell_setting, *6*, *21*, 110, *124*, **255**, *255*, *559*, *563*, 565, 567
—.entry_id, 190, *441*, **442**
—._equiv_pos_as_xyz, *21*
—.Int_Tables_number, 190, *441*, **442**
—._Int_Tables_number, 110, **255**, *563*
—.space_group_name_H-M, *5*, 190, *441*, **442**
—._space_group_name_H-M, *6*, *21*, 110, 111, *124*, *255*, **256**, *559*, *563*, 565–567
—.space_group_name_Hall, 190, **442**
—._space_group_name_Hall, 110, *255*, **256**, *559*, *563*, 567
SYMMETRY_EQUIV, 110, 111, 190, **256**, 289, **442**
—.id, **442**, *442*
—.pos_as_xyz, 190, **442**, *442*
—._pos_as_xyz, *6*, 85, 111, *111*, *124*, **256**, *256*, *559*, *563*, 565, 567
—._pos_site_id, 109, 111, *111*, *124*, **256**, *256*

_type, *see* DDL1
_type_conditions, *see* DDL1
_type_construct, *see* DDL1

_units, *see* DDL1
_units_detail, *see* DDL1

VALENCE_PARAM, 111, 190, **256**, **442**
—.atom_1, 190, **442**, *442*
—._atom_1, 111, *112*, **256**, *256*
—.atom_1_valence, 190, **442**, *442*
—._atom_1_valence, 111, *112*, **256**, *256*
—.atom_2, 190, **442**, *442*
—._atom_2, 111, *112*, **256**, *256*
—.atom_2_valence, 190, *442*, **443**
—._atom_2_valence, 111, *112*, **256**, *256*
—.B, 190, *442*, **443**
—._B, 111, *112*, **256**, *256*

VALENCE_PARAM
—.details, 190, *442*, **443**
—._details, 111, *112*, **256**, *256*
—.id, 190, **443**
—._id, 111, *256*, **257**
VALENCE_PARAM
—.ref_id, 190, *442*, **443**
—._ref_id, 111, *112*, *256*, **257**
—.Ro, 190, *442*, **443**
—._Ro, 111, *112*, *256*, **257**
VALENCE_REF, 111, 190, 256, **257**, 442, **443**
—.id, 190, *442*, **443**
—._id, 111, *112*, *256*, **257**
—.reference, 190, *442*, **443**
—._reference, 111, *112*, *256*, **257**